Raúl Toral and
Pere Colet

Stochastic Numerical Methods

Related Titles

Dubbeldam, J., Green, K., Lenstra, D. (eds.)

The Complexity of Dynamical Systems

A Multi-disciplinary Perspective

2011
Print ISBN: 978-3-527-40931-0
ISBN: 978-0-470-05480-2

Huynh, H.H., Soumaré, I.I., Lai, V.V.

Stochastic Simulation and Applications in Finance with MATLAB Programs

2008
ISBN: 978-0-470-72538-2
Also available in digital formats.

Yates, R.D., Goodman, D.J.

WIE Probability and Stochastic Processes

A Friendly Introduction for Electrical and Computer Engineers, 2nd Edition, International Edition

2 Edition
2005
ISBN: 978-0-471-45259-1

Gilat, A.

MATLAB

An Introduction with Applications 2nd Edition

2 Edition
2005
ISBN: 978-0-471-69420-5
Also available in digital formats.

Iosifescu, M., Limnios, N., Oprisan, G.

Introduction to Stochastic Models

2009
ISBN: 978-1-848-21057-8
Also available in digital formats.

Mahnke, R., Kaupuzs, J., Lubashevsky, I.

Physics of Stochastic Processes

How Randomness Acts in Time

2009
ISBN: 978-3-527-40840-5
Also available in digital formats.

Raúl Toral and
Pere Colet

Stochastic Numerical Methods

An Introduction for Students and Scientists

WILEY-VCH
Verlag GmbH & Co. KGaA

Authors

Prof. Raúl Toral
IFISC (Institute for Cross-disciplinary Physics and Complex Systems)
CSIC-Universitat Illes Balears
Palma de Mallorca
Spain

Prof. Pere Colet
IFISC (Institute for Cross-disciplinary Physics and Complex Systems)
CSIC-Universitat Illes Balears
Palma de Mallorca
Spain

Cover
The cover figure aims at exemplifying the random movement of Brownian particles in a potential landscape.

Library of Congress Card No.: applied for

British Library Cataloguing-in-Publication Data
A catalogue record for this book is available from the British Library.

Bibliographic information published by the Deutsche Nationalbibliothek
The Deutsche Nationalbibliothek lists this publication in the Deutsche Nationalbibliografie; detailed bibliographic data are available on the Internet at <http://dnb.d-nb.de>.

Print ISBN: 978-3-527-41149-8
ePDF ISBN: 978-3-527-68313-0
ePub ISBN: 978-3-527-68312-3
Mobi ISBN: 978-3-527-68311-6
oBook ISBN: 978-3-527-68314-7

Cover-Design
Adam-Design, Weinheim, Germany
Typesetting Laserwords Private Limited, Chennai, India
Printing and Binding Markono Print Media Pte Ltd, Singapore

Printed on acid-free paper

Dedicated to the memory of my father, from whom I learnt many important things not covered in this book.

Raúl Toral

Dedicated to my parents and to my wife for their support in different stages of my life.

Pere Colet

Contents

Preface

This book deals with numerical methods that use, in one way or another, concepts and ideas from probability theory and stochastic processes. This does not mean that their range of validity is limited to these fields, as there are many problems of a purely deterministic nature that can be tackled with these methods, among them, most noticeable, being the calculation of high-dimensional sums and integrals. The material covered in this book has grown out of a series of master courses taught by the authors in several universities and summer schools including, in particular, the Master in Physics of Complex Systems organized by IFISC (Instituto de Física Interdisciplinar y Systemas Complejos). It is aimed, then, at postgraduate students of almost any scientific discipline and also at those senior scientists who, not being familiar with the specificities of the numerical methods explained here, would like to get acquainted with the basic stochastic algorithms because they need them for a particular application in their field of research. The methods split naturally in three big blocks: sampling by Monte Carlo (Chapters 2–5 and 10), generation of trajectories of stochastic differential equations (Chapters 6, 7, and 11), and numerical solutions to master equations (Chapters 8 and 9), although they are intertwined in many occasions and we have tried to highlight those connections whenever they appear.

It has been our intention to keep the contents of the book self-contained. Hence, no previous knowledge of the subject is assumed by the reader. This is strictly true insofar as the numerical algorithms are concerned, but we have also included some chapters of a more theoretical nature where we summarize what the reader should know before facing the numerical chapters. These are: Chapter 1, a summary of probability concepts including the theory of Markov chains; Chapter 6, with a brief introduction to stochastic processes and stochastic differential equations, the concepts of white and colored noise, etc.; and Chapter 8, where we present briefly the master equations and some analytical tools for their analysis. The reader who is not familiar with any of these topics will find here the necessary theoretical concepts to understand the numerical methods, but if the reader wants to get a deeper knowledge of the theory, he or she might find it necessary to delve into the more advanced books mentioned in the bibliography section. Nevertheless, we have tried to keep the bibliographic references to a minimum, rather than include

here and there numerous references to the different authors who have contributed to some aspect of the methods explained in the book. We feel that a book with a clear pedagogical orientation, as this one, is different from a review, and the reader should not be distracted by an excess of references and we apologize to those authors who, having contributed to the field, do not find their work cited here. The basic bibliographic sites as well as suggestions for further reading have been included at the end of each chapter.

The goal of the book is to teach the reader to program the numerical algorithms to do different tasks. To this end, we have included more or less complete pieces of computer code. They are written in Fortran but they are, in the vast majority of cases, simple enough that they can be understood by anyone with a basic knowledge of programming in any language. The book is not intended to teach programming or code optimization. Sometimes we provide a full program, sometimes just some relevant lines of code. In any event, we do consider the lines of code to be part of the text, an important part that has to be read and analyzed in detail. Our experience indicates that full understanding is only achieved when one does not only know what he or she wants to do but also how this is implemented practically, and we do recommend the reader to implement and execute the computer programs along with the reading of the text. In particular, the section of the code that we consider absolutely essential has been framed in a box. It is not possible to reach a good understanding of the numerical algorithms without the understanding of the lines of code in the boxes.

We have included some exercises to complement the theory of the algorithms explained at the end of each chapter. We have not used the exercises to introduce difficult or more advanced topics but the exercises are, in general, of the level of the course and are given here so that the reader can test his or her level of comprehension of the algorithms and theory of the main text.

Some material is left for the appendices. They cover either standard material (generation of uniform random numbers, calculation of the correlation time of a series and discrete Fourier transforms), some more specialized topics (generation of Gaussian fields with a given correlation function, extrapolation techniques), or an introduction to more advanced simulation methods (collective algorithms, multicanonical simulations).

As we have said, the book is addressed to scientists of many disciplines, and beyond the general framework we have included only a few specific applications. In particular, in Chapter 5, we explain how to use the Monte Carlo sampling to derive properties of physical systems near a phase transition. The reason for the inclusion of this chapter is twofold. Historically, the field of phase transitions has made extensive use of the Monte Carlo methods (to the extent that some people might wrongly think that this is the only field of application). Moreover, it is the field of expertise of the authors, and we felt more confident explaining in detail how one can use the Monte Carlo sampling to derive precisely the equation of state and the critical exponents of some model systems of interest in Statistical Mechanics, including the Ising and scalar models, than other examples. Nevertheless,

extensions of, for instance, the Ising model are now being used in fields as distant as sociology or economics, and we hope that the reader not particularly interested in the physical applications will still find some useful aspects of this chapter.

Finally, we would like to thank all the colleagues and students who have helped us to improve this book. In particular, Dr. Emilio Hernández–Garcia read and gave us valuable suggestions for Chapters 8 and 9.

1
Review of probability concepts

In this chapter, we give a brief summary of the main concepts and results on probability and statistics which will be needed in the rest of the book. Readers who are familiar with the theory of probability might not need to read this chapter in detail, but we urge them to check that effectively this is the case.

1.1
Random Variables

In most occasions, we cannot predict with absolute certainty the outcome of an experiment (otherwise it might not be necessary to perform the experiment). We understand here the word ''experiment'' in a broad sense. We can count the number of electrons emitted by a β-radioactive substance in a given time interval, determine the time at which a projectile hits its target or a bus reaches the station, measure an electron's spin, toss a coin and look at the appearing side, or have a look through the window to observe whether it rains or not. We will denote by E the set of possible results of the experiment. For the β-radioactive substance, $E = \{0, 1, 2, \dots\}$ is the set of natural numbers $\mathbb{N}$; the hitting times of the projectile or the arrival times of the bus (in some units) both belong to the set of real numbers $E = \mathbb{R}$; the possible outcomes of a measure of an electron's spin are $E = \{-\hbar/2, \hbar/2\}$; when tossing a dice, the possible results are $E = \{\textbf{heads}, \textbf{tails}\}$; and, finally, for the rain observation the set of results is $E = \{\textbf{yes}, \textbf{no}\}$. In all these cases, we have no way (or no effective way) of knowing *a priori* which one of the possible outcomes will be observed. Hence, we abandon the deterministic point of view and adopt a ''probabilistic'' description in which subsets of results (which are called ''events'') are assigned a number measuring their likelihood of appearance. The ''theory of probability'' is the branch of mathematics that allows us to perform such an assignation in a logically consistent way and compatible with our intuition of how this likelihood of events should behave.

It is useful for the theory to consider that the set of results contains only numbers. In this way, we can use the rules of calculus (add, multiply, differentiate, integrate, etc.). If the results themselves are numbers (case of counting the number of electrons, determine the time of impact of the projectile, etc.), this requires no

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

special consideration. In other cases (to observe whether it rains or not), we need to label each result with a number. This assignation is arbitrary, but usually it responds to some logics of the problem under consideration. For instance, when tossing a coin, it might be that we win €1 every time heads show up and we lose €1 when tails appear. The "natural" identification is +1 for heads and −1 for tails. This assignation of a number to the result of an experiment is called a "random variable." Random variables are, hence, an application of the set of results to the set of real numbers. This application maps each result of the experiment $\xi \in E$ into one, and only one, number. The application need not be one-to-one, but it can be many-to-one. For instance, if the experiment is to extract cards from a shuffled deck, we could assign +2 to all hearts cards, +1 to spades, and 0 to diamonds and clubs. It is customary to denote random variables by using a "hat" on top of its name, say $\hat{\mathbf{x}}$, or $\hat{\mathbf{y}}$, or whatever name we choose for it. If we choose the name $\hat{\mathbf{x}}$, the number associated with the result ξ is $\hat{\mathbf{x}}(\xi) \in \mathbb{R}$. This distinction between the result of the experiment and the real number associated with it is important from the mathematical point of view, but in many cases of interest they both coincide because the result of the experiment is already a real number, $\xi = x$, and it is natural to define $\hat{\mathbf{x}}(x) = x$ in a somewhat redundant notation.

In summary, a random variable $\hat{\mathbf{x}}$ is a real number that is obtained as a result of an experiment.

The next step in the theory is to assign numbers called "probabilities" to the possible results of the experiment or, equivalently, to the different values of the random variable. The assignation should match our *a priori* expectations (if any) about the likelihood (expected frequency of appearance) of the different outcomes. For instance, when tossing a coin, it is natural (but not necessarily useful or convenient) to assign a probability equal to 1/2 to the appearance of heads, such that $P(\mathbf{heads}) = 1/2$ or, equivalently, to the random variable $\hat{\mathbf{x}}(\mathbf{heads})$ taking the value +1 (as assigned arbitrarily before), $P(\hat{\mathbf{x}} = +1) = 1/2$. The assignation of probabilities to events might follow some physical law (as in the case of the radioactive substance, the Boltzmann law for the distribution of energies, or the quantum mechanical postulates), might come after some lengthy calculation (the probability of rain tomorrow), or might follow other arguments such as symmetry (the probability of heads is equal to 1/2), Jayne's principle (based on the extremization of the information function), and so on; however, whatever its origin, we consider the assignation to be known. A typical consequence of the theory is the calculation of probabilities for more or less complicated events: What is the probability of obtaining five heads in a row if we toss a coin 10 times? What is the probability that the next emission of an electron by the radioactive substance occurs in the next 10 ms? and so on.

In practice, the assignation of probabilities to values of the random variable is performed differently if the random variable is continuous (i.e., it can take continuous values in a given interval $\hat{\mathbf{x}} \in (\alpha, \beta)$ where α can also be $-\infty$ or β can be $+\infty$) or discrete (can take only a finite or infinite numerable set of values $\hat{\mathbf{x}} \in \{x_1, x_2, x_3, \dots\}$). For example, the random variable counting the number of times a coin must be tossed before heads appear can take an infinite numerable

set of values $\{1, 2, 3, \ldots\}$. The time at which the daily bus reaches the station can take continuous values in a finite interval $(0, 24)$ h.

For a discrete random variable taking values $\hat{\mathbf{x}} \in \{x_1, x_2, x_3, \ldots\}$, we assign to each value x_i its probability $p_i = P(\hat{\mathbf{x}} = x_i)$ such that the following two conditions, namely nonnegativity and normalization, are satisfied:

$$p_i \geq 0, \quad \forall i, \tag{1.1}$$

$$\sum_{\forall i} p_i = 1. \tag{1.2}$$

One can check that the assigned probabilities p_i are consistent with the actual results of the experiment. For instance, quantum mechanics might predict that, in a given experiment with an electron's spin, the random variable $\hat{\mathbf{x}}$ takes the value $x_1 = +\hbar/2$ with probability $p_1 = 1/3$ and the value $x_2 = -\hbar/2$ with probability $p_2 = 2/3$. To check whether this prediction is correct, one repeats the experiment M (a large number) times and computes the frequency $f_i = n_i/M$, n_i being the number of times that result x_i appears, and checks whether f_1 is close to $1/3$ and f_2 to $2/3$. If they are not, then the predictions of the theory or the implementation of the experiment are wrong.[1)]

For a continuous random variable $\hat{\mathbf{x}}$, we assign, instead, a probability to the random variable taking a value in a finite interval $[a, b]$ as

$$P(\hat{\mathbf{x}} \in [a, b]) = \int_a^b f_{\hat{\mathbf{x}}}(x)\, dx. \tag{1.3}$$

Here, $f_{\hat{\mathbf{x}}}(x)$ is known as the *probability density function* of the random variable $\hat{\mathbf{x}}$, or pdf for short. It is one of the most important concepts in the theory of random variables. To be able to consider $f_{\hat{\mathbf{x}}}(x)$ as a *bona fide* pdf, it must satisfy the nonnegativity and normalization conditions:

$$f_{\hat{\mathbf{x}}}(x) \geq 0, \tag{1.4}$$

$$\int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}}(x)\, dx = 1. \tag{1.5}$$

The interpretation of the pdf is that, in the limit $dx \to 0$, $f_{\hat{\mathbf{x}}}(x)\, dx$ gives the probability that the random variable $\hat{\mathbf{x}}$ takes values between x and $x + dx$, that is

$$P(x < \hat{\mathbf{x}} \leq x + dx) = f_{\hat{\mathbf{x}}}(x)\, dx. \tag{1.6}$$

In this way, the probability that the random variable $\hat{\mathbf{x}}$ takes a value within an arbitrary region $\Omega \subset \mathbb{R}$ of the real numbers is given by the integral of the pdf over that region:

$$P(\hat{\mathbf{x}} \in \Omega) = \int_\Omega f_{\hat{\mathbf{x}}}(x)\, dx. \tag{1.7}$$

Note that $f_{\hat{\mathbf{x}}}(x)$ has units of the inverse of the units of x, and it is not limited to taking values smaller than or equal to 1. A pdf governing the probability

1) An important issue in probability theory is to be able to conclude whether the observed frequencies f_i are indeed compatible, *within unavoidable statistical errors*, with the postulated probabilities p_i.

of the next emission of an electron by a β-radioactive substance has units of inverse of time, or T^{-1}. A pdf can be computed from the experimental data. We first generate M data of the random variable $\hat{\mathbf{x}}$ by repeating the experiment M times and recording the outcomes $\{x_1, x_2, \dots, x_M\}$. We choose an interval Δx and count the number of times $n(x, x+\Delta x)$ in which the random variable has taken values in the interval $(x, x+\Delta x)$. According to the interpretation of $f_{\hat{\mathbf{x}}}(x)$, it is $f_{\hat{\mathbf{x}}}(x) \approx n(x, x+\Delta x)/(M\Delta x)$, from which $f_{\hat{\mathbf{x}}}(x)$ can be estimated. A good estimate for $f_{\hat{\mathbf{x}}}(x)$ requires M to be large and Δx to be small. Again, if the estimated $f_{\hat{\mathbf{x}}}(x)$ is not equal to the theoretical prediction, then something is wrong with the theory or with the experiment.

Further calculations can be simplified if one introduces the cumulative distribution function or cdf, $F_{\hat{\mathbf{x}}}(x)$, as

$$F_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{x} f_{\hat{\mathbf{x}}}(x')\, dx'. \tag{1.8}$$

From this definition, it follows that the cdf $F_{\hat{\mathbf{x}}}(x)$ is the probability that the random variable $\hat{\mathbf{x}}$ takes values less or equal than x:

$$P(\hat{\mathbf{x}} \le x) = F_{\hat{\mathbf{x}}}(x) \tag{1.9}$$

and that

$$P(x_1 < \hat{\mathbf{x}} \le x_2) = F_{\hat{\mathbf{x}}}(x_2) - F_{\hat{\mathbf{x}}}(x_1) \tag{1.10}$$

which is a relation that will be useful later. The following general properties arise from the definition, the nonnegativity (1.4), and the normalization condition (1.5) of the pdf $f_{\hat{\mathbf{x}}}(x)$:

$$F_{\hat{\mathbf{x}}}(x) \ge 0, \tag{1.11}$$

$$\lim_{x\to-\infty} F_{\hat{\mathbf{x}}}(x) = 0, \tag{1.12}$$

$$\lim_{x\to+\infty} F_{\hat{\mathbf{x}}}(x) = 1, \tag{1.13}$$

$$x_2 > x_1 \Rightarrow F_{\hat{\mathbf{x}}}(x_2) \ge F_{\hat{\mathbf{x}}}(x_1). \tag{1.14}$$

The last property tells us that $F_{\hat{\mathbf{x}}}(x)$ is a nondecreasing function of its argument.

If $f_{\hat{\mathbf{x}}}(x)$ is piecewise continuous, then the probability of the random variable $\hat{\mathbf{x}}$ taking a particular value x is equal to zero, as it must be understood as the limit $P(\hat{\mathbf{x}} = x) = \lim_{\Delta x\to 0} \int_x^{x+\Delta x} f_{\hat{\mathbf{x}}}(x)\, dx = 0$. It is possible to treat discrete variables in the language of pdfs if we use the "Dirac-delta function" $\delta(x)$. This mathematical object is not a proper function, but it can be understood[2] as the limit of a succession of functions $\delta_n(x)$ such that $\delta_n(x)$ decays to zero outside a region of width $1/n$ around $x = 0$ and has a height at $x = 0$ or order n such that the integral $\int_{-\infty}^{\infty} dx\, \delta_n(x) = 1$. There are many examples of such functions: for instance, $\delta_n(x) = n/\sqrt{2\pi}\mathrm{e}^{-n^2x^2/2}$, or $\delta_n(x) = \begin{cases} 0, & x \notin (-1/n, 1/n) \\ n(1-n|x|), & x \in (-1/n, 1/n) \end{cases}$, see Figure 1.1. The detailed shape is

2) Another way, more rigorous from the mathematical point of view, to introduce the Dirac-delta is by the use of distribution theory, but this is beyond the scope of this book.

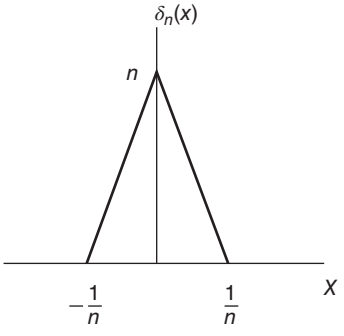


Figure 1.1 Function $\delta_n(x)$. It has the property that $\int_{-\infty}^{\infty} dx\delta_n(x) = 1$ and, when $n \to \infty$, it tends to the delta function $\delta(x)$.

not important. What matters is that, in the limit $n \to \infty$, these functions tend to yield a nonzero value only at $x = 0$ while keeping their integral over all $\mathbb{R}$ constant. As, for an arbitrary function $f(x)$, we have

$$\lim_{n\to\infty} \int_{-\infty}^{\infty} dx\, \delta_n(x) f(x) = f(0) \tag{1.15}$$

and so we can (in a nonrigorous way) exchange the limit and the integral and understand the Dirac-delta function as satisfying

$$\delta(x) = 0, \quad \text{if } x \neq 0, \tag{1.16}$$

$$\int_{-\infty}^{\infty} dx f(x)\delta(x) = f(0). \tag{1.17}$$

When the random variable takes a discrete (maybe infinite numerable) set of values $\hat{\mathbf{x}} \in \{x_1, x_2, x_3, \dots\}$ such that the value x_i has probability p_i, then the pdf can be considered as a sum of Dirac-delta functions:

$$f_{\hat{\mathbf{x}}}(x) = \sum_{\forall i} p_i \delta(x - x_i) \tag{1.18}$$

because now $P(\hat{\mathbf{x}} = x_i) = \lim_{\Delta x\to 0} \int_{x_i-\Delta x}^{x_i+\Delta x} f_{\hat{\mathbf{x}}}(x)\, dx = p_i$. The corresponding cumulative function is a sum of Heaviside step functions:

$$F_{\hat{\mathbf{x}}}(x) = \sum_{\forall i} p_i \theta(x - x_i) \tag{1.19}$$

with the usual definition

$$\theta(x) = \begin{cases} 0, & x < 0, \\ 1, & x \geq 0. \end{cases} \tag{1.20}$$

1.2 Average Values, Moments

As a random variable $\hat{\mathbf{x}}$ assigns a real number $\hat{\mathbf{x}}(\xi)$ to the result of the experiment ξ, it is possible to use a given real function $G(x)$ to define a new random variable $\hat{\mathbf{G}}$ as $\hat{\mathbf{G}}(\xi) = G(\hat{\mathbf{x}}(\xi))$. One defines the average or expected value $E[\hat{\mathbf{G}}]$ of this random variable as

$$E[\hat{\mathbf{G}}] = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}}(x) G(x)\, dx. \tag{1.21}$$

The alternative notations $\langle \hat{\mathbf{G}} \rangle$ or simply $E[G]$ and $\langle G \rangle$ are very common and will also be used in this book. In particular, for a discrete random variable with pdf given by (1.18), the average value is

$$E[\hat{\mathbf{G}}] = \sum_{\forall i} p_i G(x_i). \tag{1.22}$$

Some important expected values are as follows:

- Mean or average value of the random variable: $\mu[\hat{\mathbf{x}}] = E[\hat{\mathbf{x}}]$;
- Moments of order n: $E[\hat{\mathbf{x}}^n]$;
- Central moments of n: $E[(\hat{\mathbf{x}} - \mu[\hat{\mathbf{x}}])^n]$;
- Variance: $\sigma^2[\hat{\mathbf{x}}] = E[(\hat{\mathbf{x}} - \mu[\hat{\mathbf{x}}])^2] = E[\hat{\mathbf{x}}^2] - E[\hat{\mathbf{x}}]^2$. The value $\sigma[\hat{\mathbf{x}}]$ is the standard deviation of the random variable $\hat{\mathbf{x}}$.

If two random variables $\hat{\mathbf{y}}$ and $\hat{\mathbf{x}}$ are related by a known function $\hat{\mathbf{y}} = y(\hat{\mathbf{x}})$, then their respective pdfs are also related.

$$f_{\hat{\mathbf{y}}}(y) = \sum_{\mu} \frac{f_{\hat{\mathbf{x}}}(x_\mu)}{\left| \dfrac{dy}{dx} \right|_{x=x_\mu}} \tag{1.23}$$

where x_μ are the solutions of the equation $y = y(x)$. For instance, if the change is $\hat{\mathbf{y}} = \hat{\mathbf{x}}^2$, then the equation $y = x^2$ has no solutions for $y < 0$ and two solutions $x_1 = +\sqrt{y}$, $x_2 = -\sqrt{y}$ for $y \geq 0$, and the pdf for $\hat{\mathbf{y}}$ is

$$f_{\hat{\mathbf{y}}}(y) = \begin{cases} 0, & y < 0, \\ \dfrac{f_{\hat{\mathbf{x}}}(\sqrt{y}) + f_{\hat{\mathbf{x}}}(-\sqrt{y})}{2\sqrt{y}}, & y \geq 0. \end{cases} \tag{1.24}$$

1.3 Some Important Probability Distributions with a Given Name

1.3.1 Bernoulli Distribution

The so-called Bernoulli distribution describes a binary experiment in which only two exclusive options are possible: A or $\overline{A}$ (“heads or tails”, “either it rains or not”), with respective probabilities p and $1 - p$, being $p \in [0, 1]$. We define the discrete

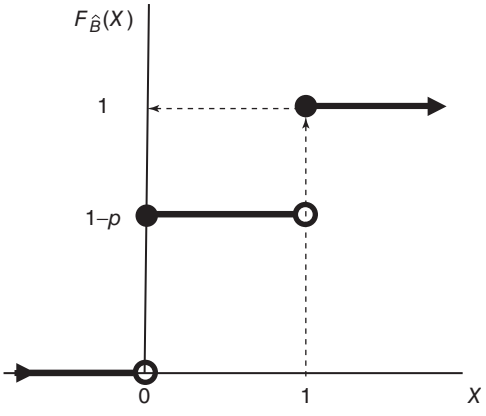


Figure 1.2 Cumulative distribution function (cdf) $F_{\hat{\mathbf{B}}}(x)$ of the Bernoulli random variable $\hat{\mathbf{B}}(p)$.

Bernoulli random variable $\hat{\mathbf{B}}$ as taking the value 1 (respectively 0) if the experiment yields A (respectively $\overline{A}$). The probabilities are

$$P(\hat{\mathbf{B}} = 1) = p, \tag{1.25}$$

$$P(\hat{\mathbf{B}} = 0) = 1 - p. \tag{1.26}$$

The mean value and variance can be computed as

$$E[\hat{\mathbf{B}}] = p, \tag{1.27}$$

$$\sigma^2[\hat{\mathbf{B}}] = p(1 - p). \tag{1.28}$$

Eventually, and when needed, we will use the notation $\hat{\mathbf{B}}(p)$ to denote a random variable that follows a Bernoulli distribution with parameter p. According to the general expression, the cdf of this random variable is $F_{\hat{\mathbf{B}}}(x) = (1 - p)\theta(x) + p\theta(x - 1)$, or

$$F_{\hat{\mathbf{B}}}(x) = \begin{cases} 0, & x < 0, \\ 1 - p, & 0 \le x < 1, \\ 1, & x \ge 1 \end{cases} \tag{1.29}$$

which is plotted in Figure 1.2.

1.3.2 Binomial Distribution

We now repeat M times the binary experiment of the previous case and count how many times A appears (independently of the order of appearance). This defines a random variable which we call $\hat{\mathbf{N}}_B$. It is a discrete variable that can take any integer value between 0 and M with probabilities

$$p(\hat{\mathbf{N}}_B = n) = \binom{M}{n} p^n (1 - p)^{M-n}. \tag{1.30}$$

$\hat{\mathbf{N}}_B$ is said to follow a binomial distribution. The mean value and the variance are given by

$$E[\hat{\mathbf{N}}_B] = Mp, \tag{1.31}$$

$$\sigma^2[\hat{\mathbf{N}}_B] = Mp(1-p). \tag{1.32}$$

We will denote by $\hat{\mathbf{N}}_B(p, M)$ a random variable that follows a binomial distribution with probability p and number of repetitions M.

1.3.3 Geometric Distribution

We consider, again, repetitions of the binary experiment, but now the random variable $\hat{\mathbf{N}}_G$ is defined as the number of times we must repeat the experiment before the result A appears (not including the case in which A does appear). This is a discrete random variable that can take any integer value $0, 1, 2, 3, \ldots$. The probability that it takes a value equal to n is

$$p(\hat{\mathbf{N}}_G = n) = (1-p)^n p, \quad n = 0, 1, 2, \ldots . \tag{1.33}$$

The mean value and variance are

$$E[\hat{\mathbf{N}}_G] = \frac{1-p}{p}, \tag{1.34}$$

$$\sigma^2[\hat{\mathbf{N}}_G] = \frac{1-p}{p^2}. \tag{1.35}$$

1.3.4 Uniform Distribution

This is our first example of a continuous random variable. We want to describe an experiment in which all possible results are real numbers within the interval (a, b) occurring with the same probability, while no result can appear outside this interval. We will use the notation $\hat{\mathbf{U}}(a, b)$ to indicate a uniform random variable in the interval (a, b). The pdf is then constant within the interval (a, b) and 0 outside it. Applying the normalization condition, it is precisely

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} \dfrac{1}{b-a}, & x \in [a, b], \\ 0, & x \notin [a, b]. \end{cases} \tag{1.36}$$

The cumulative function is

$$F_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < a, \\ \dfrac{x-a}{b-a}, & a \le x < b, \\ 1, & x \ge b. \end{cases} \tag{1.37}$$

These two functions are plotted in Figure 1.3.

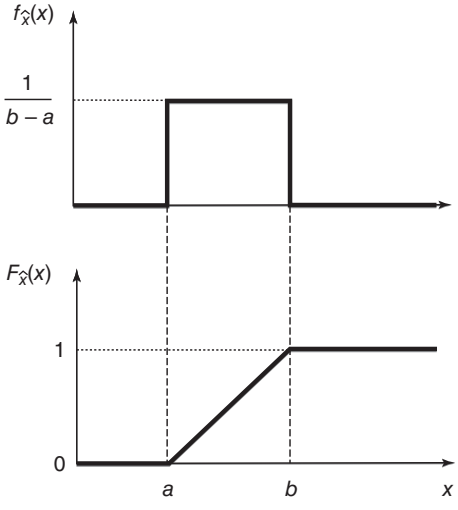


Figure 1.3 Top: Probability density function (pdf) $f_{\hat{\mathbf{x}}}(x)$ of the $\hat{\mathbf{U}}(a,b)$ distribution uniformly distributed in the interval (a,b). Bottom: The corresponding cumulative distribution function (cdf) $F_{\hat{\mathbf{x}}}(x)$.

The mean value and variance are

$$E[\hat{\mathbf{x}}] = \frac{a+b}{2}, \tag{1.38}$$

$$\sigma^2[\hat{\mathbf{x}}] = \frac{(b-a)^2}{12}. \tag{1.39}$$

The uniform distribution $\hat{\mathbf{U}}(0,1)$ appears in a very important result. Let us consider an arbitrary random variable $\hat{\mathbf{x}}$ (discrete or continuous) whose cdf is $F_{\hat{\mathbf{x}}}(x)$, and let us define the new random variable $\hat{\mathbf{u}} = F_{\hat{\mathbf{x}}}(\hat{\mathbf{x}})$. We will prove that $\hat{\mathbf{u}}$ is a $\hat{\mathbf{U}}(0,1)$ variable.

The proof is as follows: Let us compute the cdf of $\hat{\mathbf{u}}$ starting from its definition

$$F_{\hat{\mathbf{u}}}(u) = P(\hat{\mathbf{u}} \le u) = P(F_{\hat{\mathbf{x}}}(\hat{\mathbf{x}}) \le u). \tag{1.40}$$

As $F_{\hat{\mathbf{x}}}(x) \in [0,1]$, the condition $F_{\hat{\mathbf{x}}}(\hat{\mathbf{x}}) \le u$ requires necessarily $u \ge 0$, so $F_{\hat{\mathbf{u}}}(u) = 0$ if $u < 0$. If, on the other hand, $u > 1$, then the condition $F_{\hat{\mathbf{x}}}(\hat{\mathbf{x}}) \le u$ is always satisfied and its probability is 1, or $F_{\hat{\mathbf{u}}}(u) = 1$ if $u \ge 1$. Finally, for $u \in (0,1)$, the condition $F_{\hat{\mathbf{x}}}(\hat{\mathbf{x}}) \le u$ is equivalent to $\hat{\mathbf{x}} \le F_{\hat{\mathbf{x}}}^{-1}(u)$, as the function $F_{\hat{\mathbf{x}}}(x)$ is a nondecreasing function. This gives

$$F_{\hat{\mathbf{u}}}(u) = P(\hat{\mathbf{x}} \le F_{\hat{\mathbf{x}}}^{-1}(u)) = F_{\hat{\mathbf{x}}}(F_{\hat{\mathbf{x}}}^{-1}(u)) = u. \tag{1.41}$$

Summing up,

$$F_{\hat{\mathbf{u}}}(u) = \begin{cases} 0, & u < 0, \\ u, & 0 \le u \le 1, \\ 1, & u > 1, \end{cases} \tag{1.42}$$

which is nothing but the cdf of a uniform random variable $\hat{\mathbf{U}}(0,1)$.

1.3.5 Poisson Distribution

Let us consider the binomial distribution in the limit of infinitely many repetitions M. If we take the double limit $M \to \infty$, $p \to 0$ but keeping $Mp \to \lambda$, a finite value, the binomial distribution $\hat{\mathbf{N}}_B(p)$ tends to the so-called Poisson distribution $\hat{\mathbf{P}}(\lambda)$. With the help of the Stirling approximation $m! \approx m^m \mathrm{e}^{-m}\sqrt{2\pi m}$, which is valid in the limit $m \to \infty$, we can prove, starting from (1.30), the following expression for the probabilities of the Poisson distribution:

$$P(\hat{\mathbf{P}} = n) = \mathrm{e}^{-\lambda}\frac{\lambda^n}{n!}, \qquad n = 0, 1, \ldots, \infty. \tag{1.43}$$

The Poisson distribution is one of the most important distributions in nature, probably second only to the Gaussian distribution (to be discussed later). The Poisson distribution has both mean and variance equal to the parameter λ:

$$E[\hat{\mathbf{P}}] = \sigma^2[\hat{\mathbf{P}}] = \lambda \tag{1.44}$$

which is a typical property that characterizes the Poisson distribution.

We can think of the Poisson distribution simply as a convenient limit that simplifies the calculations in many occasions. For instance, the probability that a person was born on a particular day, say 1 January, is $p = 1/365$, approximately.[3] Imagine that we have now a large group of $M = 500$ people. What is the probability that exactly three people were born on 1 January? The correct answer is given by the binomial distribution by considering the events A="being born on 1 January" with probability $p = \frac{1}{365}$ and $\overline{A}$="not being born on 1 January" with probability $1 - p = \frac{364}{365}$:

$$P(\hat{\mathbf{N}}_B = 3) = \binom{500}{3}\left(\frac{1}{365}\right)^3\left(\frac{364}{365}\right)^{497} = 0.108919\ldots \tag{1.45}$$

As p is small and M large, we might find it justified to use the Poisson approximation, $\lambda = pM \approx 500/365 = 1.37$, to obtain

$$P(\hat{\mathbf{P}} = 3) = \mathrm{e}^{-1.37}\frac{1.37^3}{3!} = 0.108900\ldots \tag{1.46}$$

which is good enough. Let us compute now, using this limit, the probability that at least two persons were born on 11 May.

$$\begin{aligned} P(\hat{\mathbf{P}} \geq 2) &= 1 - P(\hat{\mathbf{P}} \leq 1) = 1 - P(\hat{\mathbf{P}} = 0) - P(\hat{\mathbf{P}} = 1) \\ &= 1 - \mathrm{e}^{\lambda} - \lambda\mathrm{e}^{\lambda} = 0.3977\ldots \end{aligned} \tag{1.47}$$

which is to be compared with the exact result $1 - P(\hat{\mathbf{N}}_B = 0) - P(\hat{\mathbf{N}}_B = 1) = 1 - \binom{500}{0}\left(\frac{1}{365}\right)^0\left(\frac{364}{365}\right)^{500} - \binom{500}{1}\left(\frac{1}{365}\right)^1\left(\frac{364}{365}\right)^{499} = 0.397895\ldots$, which is again a reasonable approximation.

There are occasions in which the Poisson limit occurs exactly. Imagine we distribute M dots randomly with a distribution $\hat{\mathbf{U}}[0, T]$, uniform in the interval

3) Neglecting leap years and assuming that all birth days are equally probable.

$[0, T]$ (we will think immediately of this as events occurring randomly in time with a uniform rate, hence the notation). We call $\omega = M/T$ the "rate" (or "frequency") at which points are distributed. We now ask the question: what is the probability that exactly k of the M dots lie in the interval $[t_1, t_1 + t] \in [0, T]$? The event A="one given dot lies in the interval $[t_1, t_1 + t]$" has probability $p = \frac{t}{T}$, whereas the event $\overline{A}$="the given dot does not lie in the interval $[t_1, t_1 + t]$" has probability $q = 1 - p$. The required probability is given by the binomial distribution, $\hat{\mathbf{B}}(p, M)$, defined by (1.30). We now make the limit $M \to \infty$, $T \to \infty$ but $\omega = M/T$ finite. This limit corresponds to the distribution in which the events occur uniformly in time with a rate (frequency) ω. As mentioned earlier, it can be proven, using Stirling's approximation, that, in this limit, the binomial distribution $\hat{\mathbf{B}}(p, M)$ tends to a Poisson distribution $\hat{\mathbf{P}}(\lambda)$, of parameter $\lambda = pM = \omega t$, finite. Let us give an example. Consider N atoms of a β-radioactive substance. Each atom emits one electron independently of the others. The probability that the given atom will disintegrate is constant with time, but it is not known which atoms will disintegrate in a given time interval. All we observe is the emission of electrons with a given rate. It is true that, as time advances, the number of atoms that can disintegrate decreases, although for some radioactive elements the decay rate is extremely slow (on the order of billions of years for the radioactive element $^{40}_{19}$K, for example). We can hence assume a constant rate ω which can be estimated simply by counting the number of electrons M emitted in a time interval T as $\omega = M/T$. Under these circumstances, the number k of electrons emitted in the time interval $[t_1, t_1 + t]$ follows a Poisson distribution of the parameter $\lambda = pM = \frac{t}{T}M = \omega t$, or

$$P(k; t) = \mathrm{e}^{-\omega t} \frac{(\omega t)^k}{k!}. \tag{1.48}$$

1.3.6 Exponential Distribution

A continuous random variable $\hat{\mathbf{x}}$ follows an exponential distribution if its pdf is

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ a\mathrm{e}^{-ax}, & x \geq 0. \end{cases} \tag{1.49}$$

The mean value and variance are

$$E[\hat{\mathbf{x}}] = \frac{1}{a}, \tag{1.50}$$

$$\sigma^2[\hat{\mathbf{x}}] = \frac{1}{a^2} \tag{1.51}$$

with $a > 0$ being a parameter.

An interesting example is related to the Poisson distribution. Consider the emission of electrons by a radioactive substance which we know is governed by the Poisson distribution for those time intervals such that the emission rate can be considered constant. Let us set our clock at $t = 0$ and then measure the time t of the first observed emission of an electron. This time is a random variable $\hat{\mathbf{t}}$ (a number

associated with the result of an experiment) and has a pdf which we call $f_{\hat{\mathbf{t}}}^{1\text{st}}(t)$. By definition, $f_{\hat{\mathbf{t}}}^{1\text{st}}(t)dt$ is the probability that the first electron is emitted during the interval $(t, t+dt)$, and, accordingly, the probability that the first electron is emitted after time t is $\int_t^\infty f_{\hat{\mathbf{t}}}^{1\text{st}}(t')dt'$. This is equal to the probability that no electrons have been emitted during $(0, t)$ or $P(0; t) = \mathrm{e}^{-\omega t}$, that is

$$\int_t^\infty f_{\hat{\mathbf{t}}}^{1\text{st}}(t')dt' = \mathrm{e}^{-\omega t},\ t \geq 0. \tag{1.52}$$

Taking the time derivate on both sides of this equation, we obtain $f_{\hat{\mathbf{t}}}^{1\text{st}}(t) = \omega \mathrm{e}^{-\omega t}$, which is valid for $t \geq 0$, the exponential distribution. Alternatively, if $\hat{\mathbf{t}}$ follows this exponential distribution, then the number of events occurring in a time interval $(0, 1)$ follows a Poisson $\hat{\mathbf{P}}(\lambda)$ distribution with $\lambda = \omega \times 1 = \omega$.

1.3.7 Gaussian Distribution

A continuous random variable $\hat{\mathbf{x}}$ follows a Gaussian distribution if its pdf is

$$f_{\hat{\mathbf{x}}}(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]. \tag{1.53}$$

The average and variance are given by

$$E[\hat{\mathbf{x}}] = \mu, \tag{1.54}$$

$$\sigma^2[\hat{\mathbf{x}}] = \sigma^2. \tag{1.55}$$

We will use the notation that $\hat{\mathbf{x}}$ is a $\hat{\mathbf{G}}(\mu, \sigma)$ random variable. The cdf is

$$F_{\hat{\mathbf{x}}}(x) = \frac{1}{2} + \frac{1}{2}\mathrm{erf}\left(\frac{x-\mu}{\sigma\sqrt{2}}\right) \tag{1.56}$$

with $\mathrm{erf}(z)$ being the error function

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-\gamma^2}\, d\gamma. \tag{1.57}$$

$f_{\hat{\mathbf{x}}}(x)$ and $F_{\hat{\mathbf{x}}}(x)$ are plotted in Figure (1.4).

Gaussian random variables are very important in practice because they appear in a large number of problems, either as an exact distribution in some limit or, simply, as providing a sufficient approximation to the real distribution. After all, it is not unusual that many distributions have a maximum value and this can in many cases be approximated by a Gaussian distribution (the so-called bell-shaped curve). One of the reasons for the widespread appearance of Gaussian distributions is the central-limit theorem, which states that the sum of a large number of independent random variables, whatever their distribution, will approach a Gaussian distribution.

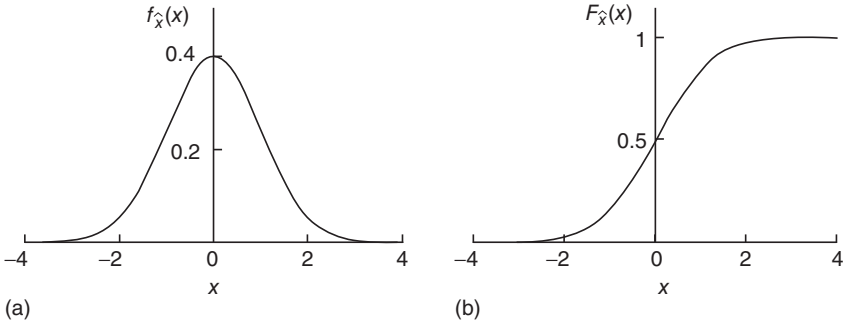


Figure 1.4 pdf and cdf of the Gaussian distribution of mean 0 and variance 1.

As a first example, one we can prove that the binomial distribution $\hat{\mathbf{N}}_B(p, M)$ tends to the Gaussian distribution $\hat{\mathbf{G}}(Mp, \sqrt{Mp(1-p)})$. More precisely

$$P(\hat{\mathbf{N}}_B = n) = \binom{M}{n} p^n (1-p)^{M-n} \approx \frac{\exp\left[-(n-Mp)^2/2Mp(1-p)\right]}{\sqrt{2\pi Mp(1-p)}}. \tag{1.58}$$

The theorem of de Moivre–Laplace, loosely speaking, establishes the equality of both sides of the previous equation in the limit $M \to \infty$ if $|n - Mp| / \sqrt{Mp(1-p)}$ remains finite. In practice, the above approximation is sufficiently good for $M \geq 100$ if $p = 0.5$ or $M \geq 1000$ if $p = 0.1$ (Figure 1.5). As can be seen in these figures, the Gaussian approximation to the binomial distribution is best around the maximum of the distribution and worsens in the tails. An equivalent way of stating the equivalence of both distributions is to define the random variable $\hat{\mathbf{x}} = \frac{\hat{\mathbf{N}}_B - Mp}{\sqrt{Mp(1-p)}}$. As $\langle \hat{\mathbf{x}} \rangle = 0$ and $\sigma[\hat{\mathbf{x}}] = 1$, it follows that $\hat{\mathbf{x}}$ obeys a Gaussian distribution $\hat{\mathbf{G}}(0, 1)$ in the limit $M \to \infty$.

The Gaussian distribution can also be obtained as the limit of the Poisson distribution for large parameter $\lambda \to \infty$. This yields a Gaussian distribution of the same mean and variance, or $\hat{\mathbf{G}}(\lambda, \sqrt{\lambda})$. Again, although the exact result refers to the limit $\lambda \to \infty$, in practice the approximation can be considered sufficient for $\lambda \geq 100$, especially around the maximum of the distribution (Figure 1.5).

1.3.8 Gamma Distribution

A random variable $\hat{\mathbf{x}}$ is said to follow a gamma distribution $\hat{\mathbf{\Gamma}}(\alpha, \theta)$ with shape α and scale parameter θ if the pdf is

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ \dfrac{x^{\alpha-1} e^{-x/\theta}}{\theta^\alpha \Gamma(\alpha)}, & x \geq 0 \end{cases} \tag{1.59}$$

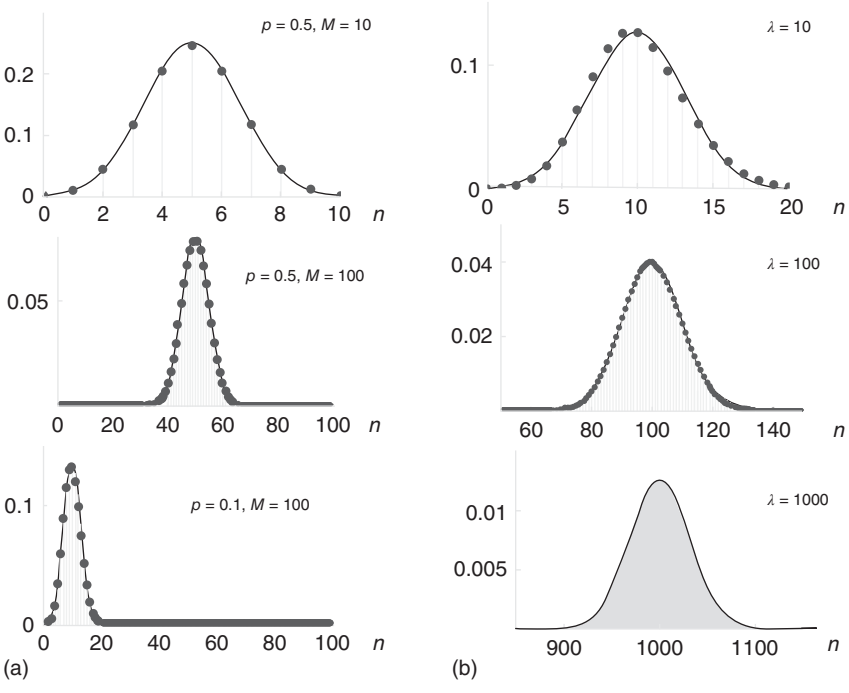


Figure 1.5 (a) Binomial distribution (dots) and its Gaussian approximation (solid line). (b) Poisson distribution (dots) and its binomial approximation (solid line).

where $\alpha > 0$, $\theta > 0$ are real numbers. Here, $\Gamma(\alpha)$ is the Gamma function which has the interesting property that $\Gamma(x+1) = x\Gamma(x)$ and that equals the factorial when the argument is an integer number $\Gamma(n+1) = n!$. The mean value and variance of this distribution are given by

$$E[\hat{\mathbf{x}}] = \alpha\theta, \tag{1.60}$$

$$\sigma^2[\hat{\mathbf{x}}] = \alpha\theta^2. \tag{1.61}$$

The shape depends on the value of α. For $0 < \alpha < 1$, it diverges at $x = 0$, while for $\alpha \geq 1$ it has a single maximum located at $x = \alpha - 1$. In Figure 1.6, we plot the gamma distribution for different values of the parameter α.

Note that, for $\theta = 1$, the Gamma and the Poisson distributions share the property that the mean value is equal to the variance. In fact, one can prove that the Poisson distribution $\hat{\mathbf{P}}(\lambda)$ can be approximated for large values of λ by the $\hat{\mathbf{\Gamma}}(\lambda, \theta = 1)$. Evidence for this is given in Figure 1.7.

1.3.9 Chi and Chi-Square Distributions

If $\alpha = k/2$, with $k \in \mathbb{N}$ and $\theta = 2$, the Gamma distribution $\hat{\mathbf{\Gamma}}(k/2, 2)$ is also called the $\hat{\boldsymbol{\chi}}^2(k)$ or chi-square distribution with k degrees of freedom. This is the distribution

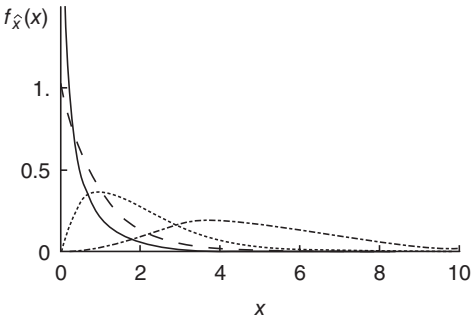


Figure 1.6 pdf for the Gamma distribution $\hat{\mathbf{\Gamma}}(\alpha, \theta = 1)$ for $\alpha = 0.5$ (solid), 1 (dashed), 2 (dotted), 5 (dot-dashed).

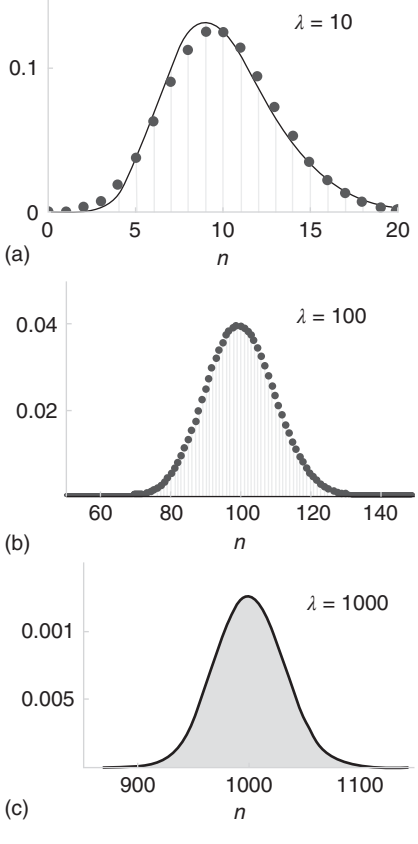


Figure 1.7 Comparison between the Poisson $\hat{\mathbf{P}}(\lambda)$ (dots) and the $\hat{\mathbf{\Gamma}}(\lambda, \theta = 1)$ (solid line) distributions for different values of the parameter λ.

followed by the sum of the squares of k independent $\hat{\mathbf{G}}(0,1)$ Gaussian variables of mean 0 and variance 1. Its square root, or chi distribution $\hat{\boldsymbol{\chi}}(k)$, follows the pdf

$$f_{\hat{\boldsymbol{\chi}}}(\chi) = \frac{2^{1-\frac{k}{2}}\chi^{k-1}e^{-\chi^2/2}}{\Gamma\left(k/2\right)} \tag{1.62}$$

with mean value and variance

$$\langle\hat{\boldsymbol{\chi}}\rangle = \sqrt{2}\frac{\Gamma((k+1)/2)}{\Gamma(k/2)}, \tag{1.63}$$

$$\sigma^2[\hat{\boldsymbol{\chi}}] = k - \langle\hat{\boldsymbol{\chi}}\rangle^2. \tag{1.64}$$

1.4 Successions of Random Variables

It is, of course, possible (and useful) to assign more that one random variable to the result of an experiment. For example, we can measure in a β-radioactive sample the time t and the speed v at which an electron is emitted; we can measure the time of arrival of the bus and the number of people in the waiting queue; we can observe whether it rains or not and measure the air temperature and pressure, and so on. In general, given an experiment, let us consider N random variables assigned to it: $(\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N)$. The joint pdf of all these random variables is a function of N real variables $f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N)$, which allows us to compute the probability that the vector of results $(\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N)$ belongs to a region Ω of $\mathbb{R}^N$ as

$$P((\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N) \in \Omega) = \int_\Omega dx_1 \dots . dx_N f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N). \tag{1.65}$$

In other words, $f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N)\, dx_1 \dots . dx_N$ is the probability that in a measurement of the N random variables the value of $\hat{\mathbf{x}}_1$ lies in $(x_1, x_1 + dx_1)$, the value of $\hat{\mathbf{x}}_2$ in $(x_2, x_2 + dx_2)$, and so on. The cdf is defined as

$$F_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\dots,\hat{\mathbf{x}}_N}(x_1, x_2, \dots, x_N) = \int_{-\infty}^{x_1} dx'_1 \int_{-\infty}^{x_2} dx'_2 \dots \int_{-\infty}^{x_N} dx'_N f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\dots,\hat{\mathbf{x}}_N}(x'_1, x'_2, \dots, x'_N). \tag{1.66}$$

We do have an intuitive idea of when some random variables can be considered independent of each other. A precise statement is that the N random variables $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N$ are defined to be statistically independent if the joint pdf factorizes as the product of pdfs for each variable, that is

$$f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N) = f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x_2) \cdots f_{\hat{\mathbf{x}}_N}(x_N). \tag{1.67}$$

The mean value of a function of N variables $G(x_1, \dots, x_N)$ is computed as

$$\langle G(x_1, \dots, x_N)\rangle = \int_{-\infty}^{\infty} dx_1 \cdots \int_{-\infty}^{\infty} dx_N G(x_1, \dots, x_N) f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N). \tag{1.68}$$

In particular, if $G(x_1, \dots, x_N) = \lambda_1 G_1(x_1) + \cdots + \lambda_N G_N(x_N)$, then

$$\langle G(x_1, \dots, x_N)\rangle = \lambda_1 \langle G_1(x_1)\rangle + \cdots + \lambda_N \langle G_N(x_N)\rangle \tag{1.69}$$

and if the random variables $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N$ are independent of each other, then

$$\sigma^2[G(x_1, \dots, x_N)] = \lambda_1^2 \sigma^2[G_1(x_1)] + \cdots + \lambda_N^2 \sigma^2[G_N(x_N)]. \tag{1.70}$$

The covariance between two of the random variables $\hat{\mathbf{x}}_i,\ \hat{\mathbf{x}}_j$ is defined as

$$C[\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j] \equiv C_{ij} \equiv \langle (\hat{\mathbf{x}}_i - \langle \hat{\mathbf{x}}_i\rangle)(\hat{\mathbf{x}}_j - \langle x_j\rangle)\rangle. \tag{1.71}$$

Note that a trivial consequence of that definition is that the matrix whose entries are the covariances is symmetrical, $C_{ij} = C_{ji}$. If the variables $\hat{\mathbf{x}}_i,\ \hat{\mathbf{x}}_j$ are statistically independent, then it is easy to verify that

$$C_{ij} = \sigma^2[\hat{\mathbf{x}}_i]\delta_{ij} \tag{1.72}$$

although the inverse statement (if $C_{ij} = \sigma^2[\hat{\mathbf{x}}_i]\delta_{ij}$ then variables $\hat{\mathbf{x}}_i,\ \hat{\mathbf{x}}_j$ are statistically independent) need not be true.

In general, the variance of the sum of two functions $G_1(x),\ G_2(x)$,

$$\sigma^2[G_1 + G_2] = \langle (G_1 + G_2)^2\rangle - \langle G_1 + G_2\rangle^2 \tag{1.73}$$

can be written as

$$\sigma^2[G_1 + G_2] = \sigma^2[G_1] + \sigma^2[G_2] + 2C[G_1, G_2] \tag{1.74}$$

with $C[G_1, G_2]$ being the covariance of G_1 and G_2.

The correlation coefficient $\rho[\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j]$ of the random variables $\hat{\mathbf{x}}_i,\ \hat{\mathbf{x}}_j$ is defined as a suitable normalization of the covariance:

$$\rho[\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j] = \frac{C[\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j]}{\sigma[\hat{\mathbf{x}}_i]\sigma[x_j]}. \tag{1.75}$$

From the definition, it follows that

$$|\rho[\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j]| \leq 1. \tag{1.76}$$

Even if there are N random variables $(\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_N)$ defined in an experiment, we can still ''forget'' about some of them and consider the pdfs of only a subset of variables, for instance, $f_{\hat{\mathbf{x}}_1}(x_1)$ or $f_{\hat{\mathbf{x}}_2\hat{\mathbf{x}}_4}(x_2, x_4)$. These are called, in this context, ''marginal'' probabilities and can be obtained by integrating out the variables that are not of interest. For example

$$f_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{\infty} dx_2 f_{\hat{\mathbf{x}}_1\hat{\mathbf{x}}_2}(x_1, x_2), \tag{1.77}$$

or

$$f_{\hat{\mathbf{x}}_2\hat{\mathbf{x}}_4}(x_2, x_4) = \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_3\, f_{\hat{\mathbf{x}}_1\hat{\mathbf{x}}_2\hat{\mathbf{x}}_3\hat{\mathbf{x}}_4}(x_1, x_2, x_3, x_4). \tag{1.78}$$

It is possible to relate the joint pdfs of two sets of related random variables: $\hat{\mathbf{y}}_1 = \gamma_1(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_n), \dots, \hat{\mathbf{y}}_n = \gamma_n(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_n)$. The result generalizing (1.23) is

$$f_{\hat{\mathbf{y}}_1,\dots,\hat{\mathbf{y}}_n}(y_1, \dots, y_n) = \sum_{\mu} \frac{f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_{1_\mu}, \dots, x_{n_\mu})}{\left| J\left(\frac{y_1,\dots,y_n}{x_1,\dots,x_n}\right)\right|_{x_i = x_{i_\mu}}} \tag{1.79}$$

where the sum runs again over all solutions of $y_1 = \gamma_1(x_1, x_2, \dots, x_n), \dots, y_n = \gamma_n(x_1, x_2, \dots, x_n)$ and J is the Jacobian matrix of coefficients $J_{ij} = \frac{\partial y_i}{\partial x_j}$.

1.5 Jointly Gaussian Random Variables

There are not many examples (besides those derived from statistically independent variables) of specific forms for a joint pdf $f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N)$. A particularly useful case is that of jointly Gaussian random variables for which the pdf is the exponential of a quadratic form, namely

$$f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_N) = \sqrt{\frac{|\mathbf{A}|}{(2\pi)^N}} \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N}(x_i - \mu_i)A_{ij}(x_j - \mu_j)\right]. \tag{1.80}$$

Here, $\mathbf{A} = \{A_{ij}\}_{i=1,\dots,N;j=1,\dots,N}$ is a symmetric matrix, and $\mu_1, \mu_2, \dots, \mu_N$ are real constants. The joint pdf has, hence, $\frac{N(N+1)}{2} + N = \frac{N(N+3)}{2}$ constants. As the pdf must be normalizable, it must go to zero as any of the variables $(x_1, \dots, x_N)$ tends to $\pm\infty$. This implies that the quadratic form in the exponential must be positive definite. A simple way of checking this out is to make sure that all eigenvalues $\lambda_1, \lambda_2, \dots, \lambda_N$ of matrix $\mathbf{A}$ are strictly positive (remember that they will be real as the matrix is symmetric). The determinant of $\mathbf{A}$ is nothing but the product of the eigenvalues $|\mathbf{A}| = \prod_{i=1}^{N} \lambda_i$. The shape of the Gaussian distribution is such that it has a maximum at the point $(\mu_1, \dots, \mu_N)$ and it decays to zero in the typical bell-shaped curve for large values of the coordinates, see Figure 1.8.

Some average values and correlations are given by

$$\langle \hat{\mathbf{x}}_i \rangle = \mu_i, \tag{1.81}$$

$$C_{ij} = (A^{-1})_{ij}. \tag{1.82}$$

Hence, the correlation matrix $\mathbf{C} = \{C_{ij}\}_{i=1,\dots,N;j=1,\dots,N}$ is the inverse of the matrix $\mathbf{A}$, $\mathbf{C} = \mathbf{A}^{-1}$. It is common, instead of writing out in full the expression (1.80), to characterize a jointly Gaussian distribution by giving the mean values μ_i and the correlations C_{ij}. In fact, most of the integrals needed in the calculations can be obtained directly from these numbers by the application of Wick's theorem. The theorem gives an expression for the calculation of averages of an even product of terms as

$$\langle (x_{i_1} - \mu_{i_1})(x_{i_2} - \mu_{i_2}) \dots (x_{i_{2n}} - \mu_{i_{2n}}) \rangle = \sum_{\text{all possible pairings}} C_{j_1 k_1} C_{j_2 k_2} \dots C_{j_n k_n}. \tag{1.83}$$

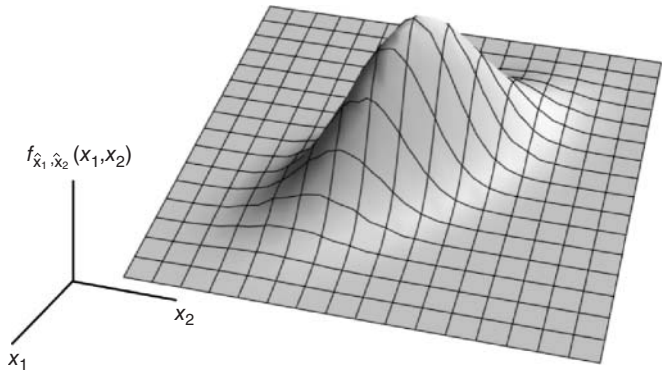


Figure 1.8 Joint Gaussian pdf in $n = 2$ variables.

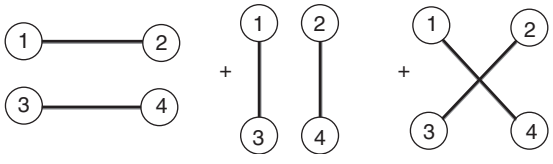


Figure 1.9 Graphs for the calculation of $\langle x_1 x_2 x_3 x_4 \rangle$ using Wick's theorem.

But, if the number of terms in the left-hand side is odd, then the average value would be 0.

Wick's theorem is better understood by specific examples. Let us take a set of four random variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \hat{\mathbf{x}}_3, \hat{\mathbf{x}}_4)$ which follow a jointly Gaussian distribution with average values $\langle x_1 \rangle = \langle x_2 \rangle = \langle x_3 \rangle = \langle x_4 \rangle = 0$ and a (symmetric) correlation matrix C_{ij}. To compute, for example, $\langle x_1 x_2 x_3 x_4 \rangle$, we need to take all possible pairings of the four numbers $1, 2, 3, 4$. They are $(1, 2)(3, 4)$, $(1, 3)(2, 4)$, and $(1, 4)(2, 3)$. This gives, according to Wick's theorem

$$\langle x_1 x_2 x_3 x_4 \rangle = C_{12} C_{34} + C_{13} C_{24} + C_{14} C_{23}. \tag{1.84}$$

It is useful to use a diagrammatic representation of this theorem. Each factor $(x_i - \mu_i)^k$ is represented by a dot with k "branches" coming out of it. All possible pairings can be realized by joining all the branches in all possible ways. Figure 1.9 exemplifies this case.

Let us consider another example for the same set of four random variables (with zero means, $\mu_i = 0$). We want to compute $\langle x_1^2 x_2 x_3 \rangle$. As this is equal to $\langle x_1 x_1 x_2 x_3 \rangle$, Wick's theorem tells us about all pairings of the variables x_1, x_1, x_2, x_3. The diagrammatic representation is shown in Figure 1.10: Or, in equation form,

$$\langle x_1^2 x_2 x_3 \rangle = C_{11} C_{23} + 2 C_{12} C_{13}. \tag{1.85}$$

The factor 2 of the second term on the right-hand side is the symmetry factor of the diagram. It corresponds to the two ways in which the branches coming out of dot number 1 can be joined to the branches of dot number 2 and dot number 3.

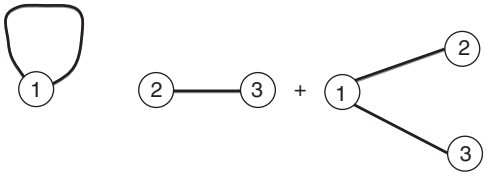


Figure 1.10 Graphs for the calculation of $\langle x_1^2 x_2 x_3 \rangle$ using Wick's theorem.

Two very important results apply to a set of N jointly Gaussian random variables:

1) The marginal pdf of a subset of m random variables is also a jointly Gaussian;
2) Defining new random variables $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n$ as linear combinations $\hat{\mathbf{y}}_i = \sum_{j=1}^{N} B_{ij}\hat{\mathbf{x}}_j$, then $\hat{\mathbf{y}}_1, \ldots, \hat{\mathbf{y}}_n$ are also jointly Gaussian variables. The new correlation matrix is $\mathbf{BCB}^{-1}$.

1.6 Interpretation of the Variance: Statistical Errors

Let us consider a random variable $\hat{\mathbf{x}}$ assigned to an experiment. In general, every time we execute an experiment and obtain a result ξ, we do not know *a priori* which numerical value, $\hat{\mathbf{x}}(\xi)$, will the random variable take (unless there exists an event with probability 1). That is why it is called a *random variable*. Imagine we do know the average value $\mu = E[\hat{\mathbf{x}}]$ and the variance $\sigma^2 = E[\hat{\mathbf{x}}^2] - E[\hat{\mathbf{x}}]^2$. Maybe this knowledge comes from some theory that provides us with the values of μ and σ. What can we say about a single outcome $\hat{\mathbf{x}}(\xi)$ of the random variable? Not much in general. But we can say something about the probability of $\hat{\mathbf{x}}(\xi)$ taking values far away from μ, the mean value. Intuitively, we expect that it is unlikely to obtain values very far away from μ. But how unlikely? Chebyshev's theorem quantifies this probability.

$$P(|\, \hat{\mathbf{x}}(\xi) - \mu \,| \geq k\,\sigma) \leq \frac{1}{k^2} \tag{1.86}$$

for an arbitrary $k > 1$. In words, the probability that a single measurement $\hat{\mathbf{x}}(\xi)$ of a random variable differs from the mean value μ by an amount larger than k times the standard deviation σ is smaller than k^{-2}. The result can be written with the equivalent expression

$$P(|\, \hat{\mathbf{x}}(\xi) - \mu \,| \leq k\,\sigma) \geq 1 - \frac{1}{k^2}. \tag{1.87}$$

For instance, if $k = 3$, it is less than $1/3^2 \approx 11\%$ probable that the result of a single experiment lies outside the interval $(\mu - 3\sigma, \mu + 3\sigma)$. In other words, we cannot predict the result of a single experiment but we can affirm with an 11% confidence (about 89 out of every 100 times we make the experiment) that it will lie in the interval $(\mu - 3\sigma, \mu + 3\sigma)$. Of course, if σ is a large number, this prediction might be useless, but the reverse is also true, that is, if σ is small, then we can

be pretty sure of the result. Imagine that the experiment is to measure the radius of one polystyrene bead taken at random from a large number we bought from a manufacturer who tells us that the average radius of the set is $\mu = 3.5$ mm and the standard deviation is $\sigma = 1.0\,\mu$m. How confident can we be that the radius of that particular bead lies in the interval (3.49, 3.51) mm? To apply Chebyshev's inequality to this data, we need to take $(3.49, 3.51) = (\mu - k\sigma, \mu + k\sigma)$ or $0.01\text{mm} = k \times 1\mu$ m or $k = 10$. This means that, on average, 1 out of $k^2 = 100$ beads will not have a radius within these limits (or, from the positive side, 99 out of 100 beads will have a radius within these limits). This interpretation of Chebyshev's theorem allows us to identify (in the precise manner defined before) the standard deviation of a distribution with the error (e.g., the uncertainty) in a *single* measurement of a random variable.

Once we have understood this, we should understand the expression

$$\hat{\mathbf{x}}(\xi) = \mu \pm \sigma \tag{1.88}$$

with $\mu = E[\hat{\mathbf{x}}]$ and $\sigma^2 = E[\hat{\mathbf{x}}^2] - E[\hat{\mathbf{x}}]^2$ as a short-hand notation of the exact statement of Chebyshev's theorem (1.86). It does not mean that experimental values $\hat{\mathbf{x}}(\xi)$ that differ from μ by an amount greater than σ cannot appear, or are forbidden; it simply means that they are unlikely. How unlikely? Exactly by $1/k^2$, with $k = \frac{|\hat{\mathbf{x}}(\xi)-\mu|}{\sigma}$.

Chebyshev's theorem is very general. It applies to any random variable whatever its pdf. In most cases, however, the Gaussian distribution is a sufficient approximation to the true (maybe unknown) distribution. In the case of a Gaussian distribution, we have the more precise result

$$P(|\,\hat{\mathbf{x}}(\xi) - \mu\,| \leq k\,\sigma) = \text{erf}\left(\frac{k}{\sqrt{2}}\right) > 1 - \frac{1}{k^2} \tag{1.89}$$

which takes the following values:

$$P(|\,\hat{\mathbf{x}}(\xi) - \mu\,| \leq \sigma) = 0.68269\ldots, \tag{1.90}$$

$$P(|\,\hat{\mathbf{x}}(\xi) - \mu\,| \leq 2\sigma) = 0.95450\ldots, \tag{1.91}$$

$$P(|\,\hat{\mathbf{x}}(\xi) - \mu\,| \leq 3\sigma) = 0.99736\ldots \tag{1.92}$$

which means that we can be certain with a 68% probability that the result of the measurement will lie in the interval $(\mu - \sigma, \mu + \sigma)$; with a 95% probability in $(\mu - 2\sigma, \mu + 2\sigma)$; and with 99.7% probability in $(\mu - 3\sigma, \mu + 3\sigma)$. Note that, if we take σ as the error of the measurement, in 32% (nearly 1/3) of the cases the observed value $\hat{\mathbf{x}}(\xi)$ will lie outside the error interval.

In most cases, one does not know the distribution function of the experiment, neither the mean μ nor the standard deviation σ. Chebyshev's theorem can be read in the inverse sense

$$\mu = \hat{\mathbf{x}}(\xi) \pm \sigma. \tag{1.93}$$

Given the result of a single measurement $\hat{\mathbf{x}}(\xi)$, this allows us to predict the value of μ within a certain interval of confidence which depends on the generally unknown standard deviation σ. However, it is clear that we cannot use this single measurement to obtain information about σ (which is ultimately related to the

dispersion in a set of measurements). To obtain estimates for both μ and σ, we have to repeat the experiment M times, each one independent of the others. We call, then, $\Xi = (\xi_1, \xi_2, \dots, \xi_M)$ the result of the experiment which consists of M independent repetitions and use some properties of the sum of random variables.

1.7 Sums of Random Variables

Let $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_M$ be independent random variables, all of them described by the same pdf $f_{\hat{\mathbf{x}}}(x)$ with mean μ and variance σ^2. The natural idea is to consider them as independent repetitions of the same experiment. Associated with the result $\Xi = (\xi_1, \xi_2, \dots, \xi_M)$, we define the random variables *sample mean* $\hat{\boldsymbol{\mu}}_M$ and *sample variance* $\hat{\boldsymbol{\sigma}}_M^2$:

$$\hat{\boldsymbol{\mu}}_M(\Xi) = \frac{1}{M}\sum_{i=1}^{M} \hat{\mathbf{x}}_i(\xi_i), \tag{1.94}$$

$$\begin{aligned}\hat{\boldsymbol{\sigma}}_M^2(\Xi) &= \frac{1}{M-1}\sum_{i=1}^{M}\left(\hat{\mathbf{x}}_i(\xi_i) - \hat{\boldsymbol{\mu}}_M\right)^2 \\ &= \frac{M}{M-1}\left(\frac{1}{M}\sum_{i=1}^{M}\hat{\mathbf{x}}_i(\xi_i)^2 - \left(\frac{1}{M}\sum_{i=1}^{M}\hat{\mathbf{x}}_i(\xi_i)\right)^2\right).\end{aligned} \tag{1.95}$$

(Note that the notation stresses the fact that both random variables depend on the number of repetitions M.) It is easy to obtain the average of these two sample random variables[4] as

$$E[\hat{\boldsymbol{\mu}}_M] = E[\hat{\mathbf{x}}_i] = \mu, \tag{1.96}$$

$$E[\hat{\boldsymbol{\sigma}}_M^2] = \sigma^2. \tag{1.97}$$

Furthermore, the variance of the sample mean is given by

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{1}{M}\sigma^2. \tag{1.98}$$

If we now repeat the experiment M times and obtain a value for $\hat{\boldsymbol{\mu}}_M(\Xi)$, we can use Chebyshev's theorem in its inverse short-hand notation (1.93) applied to the random variable $\hat{\boldsymbol{\mu}}_M$ to write $\mu = \hat{\boldsymbol{\mu}}_M(\Xi) \pm \sigma[\hat{\boldsymbol{\mu}}_M]$, or using (1.98)

$$\mu = \hat{\boldsymbol{\mu}}_M(\Xi) \pm \frac{\sigma}{\sqrt{M}}. \tag{1.99}$$

Still, we do not know the true value of σ on the right-hand side of this equation. It seems intuitive, though, given (1.97), that we can replace it by the sample variance $\sigma \approx \hat{\boldsymbol{\sigma}}_M[\Xi]$, leading to the final result

$$\mu = \hat{\boldsymbol{\mu}}_M(\Xi) \pm \frac{\hat{\boldsymbol{\sigma}}_M[\Xi]}{\sqrt{M}} \tag{1.100}$$

4) The presence of the factor $M - 1$ in (1.95) avoids its presence in (1.97).

which yields an estimate of the average value μ together with its error. As discussed before, this error has to be interpreted in the statistical sense. There is some good news here. As the sum of M independent random variables tends to a Gaussian distribution as M increases, we can take the Gaussian confidence limits and conclude that in 68% of the cases the true value of μ will lie in the interval $\left(\hat{\boldsymbol{\mu}}_M(\Xi) - \frac{\hat{\boldsymbol{\sigma}}_M[\Xi]}{\sqrt{M}}, \hat{\boldsymbol{\mu}}_M(\Xi) + \frac{\hat{\boldsymbol{\sigma}}_M[\Xi]}{\sqrt{M}}\right)$, and so on.

If we worry about the replacement $\sigma \approx \hat{\boldsymbol{\sigma}}_M[\Xi]$ in the previous formulas, we can estimate the error of this replacement (again in a statistical sense) by applying Chebyshev's theorem to the random variable $\hat{\boldsymbol{\sigma}}_M$. In the limit of large M, we assume that $\sqrt{\sum_{i=1}^{M}\left(\frac{\hat{\mathbf{x}}_i - \hat{\boldsymbol{\mu}}_M}{\sigma}\right)^2}$, and hence $\sqrt{M-1}\hat{\boldsymbol{\sigma}}_M/\sigma$ can be approximated by a $\hat{\chi}(M)$ distribution with M degrees of freedom. Using (1.63) and (1.64) and the result of Exercise 5 in the limit of large M, we conclude that $\hat{\boldsymbol{\sigma}}_M/\sigma$ follows a $\hat{\mathbf{G}}(1, 1/\sqrt{2M})$ Gaussian distribution or that $\hat{\boldsymbol{\sigma}}_M$ follows a $\hat{\mathbf{G}}(\sigma, \sigma/\sqrt{2M})$ distribution. Using again Chebyshev's theorem in its short-hand notation, we can write

$$\sigma = \hat{\boldsymbol{\sigma}}_M(\Xi) \pm \frac{\sigma}{\sqrt{2M}} \tag{1.101}$$

thereby justifying the replacement $\sigma \approx \hat{\boldsymbol{\sigma}}_M[\Xi]$, which is valid in the limit of large M. This result also leads to the standard deviation error estimation, as

$$\sigma = \hat{\boldsymbol{\sigma}}_M(\Xi) \pm \frac{\hat{\boldsymbol{\sigma}}_M(\Xi)}{\sqrt{2M}}. \tag{1.102}$$

1.8 Conditional Probabilities

As the final ingredient in this brief summary of probability theory, we review now the concept of conditional probability. For the sake of simplicity, we will consider the case of two random variables $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$, but similar ideas can be easily generalized in the case of more random variables.

The joint probability density $f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)$ is defined such that the probability that a measurement of the random variable $\hat{\mathbf{x}}$ *and* the random variable $\hat{\mathbf{y}}$ gives for each one of them a value in the interval $(x, x + dx)$ and $(y, y + dy)$, respectively, is

$$P(x < \hat{\mathbf{x}} \leq x + dx, y < \hat{\mathbf{y}} \leq y + dy) = f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)\, dx\, dy. \tag{1.103}$$

The cdf

$$F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y) = \int_{-\infty}^{x}\int_{-\infty}^{y} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(q, p)\, dq\, dp \tag{1.104}$$

is such that

$$\begin{aligned} &P(x_1 < \hat{\mathbf{x}} \leq x_2, y_1 < \hat{\mathbf{y}} \leq y_2) = \\ &F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x_2, y_2) - F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x_1, y_2) - F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x_2, y_1) + F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x_1, y_1). \end{aligned} \tag{1.105}$$

Some results follow straightforwardly from the definition:

$$\frac{\partial F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)}{\partial x} = \int_{-\infty}^{y} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,p)\, dp, \tag{1.106}$$

$$\frac{\partial F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)}{\partial y} = \int_{-\infty}^{x} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(q,y)\, dq, \tag{1.107}$$

$$\frac{\partial^2 F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)}{\partial x \partial y} = f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y). \tag{1.108}$$

The marginal probabilities are

$$f_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)\, dy, \tag{1.109}$$

$$f_{\hat{\mathbf{y}}}(y) = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)\, dx. \tag{1.110}$$

Let us recall the definition of conditional probability. For any two events A and B such that $P(B) \neq 0$, the conditional probability of A given B is defined as

$$P(A|B) = \frac{P(A \cap B)}{P(B)}. \tag{1.111}$$

This suggests the definition of the conditional distribution function

$$F_{\hat{\mathbf{y}}}(y|B) = P(\hat{\mathbf{y}} \leq y|B) = \frac{P(\hat{\mathbf{y}} \leq y, B)}{P(B)} \tag{1.112}$$

and the conditional density function

$$f_{\hat{\mathbf{y}}}(y|B) = \frac{\partial F_{\hat{\mathbf{y}}}(y|B)}{\partial y}. \tag{1.113}$$

In the particular case of the event $B = \{\hat{\mathbf{x}} \leq x\}$, we have

$$F_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} \leq x) = \frac{P(\hat{\mathbf{x}} \leq x, \hat{\mathbf{y}} \leq y)}{P(\hat{\mathbf{x}} \leq x)} = \frac{F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)}{F_{\hat{\mathbf{x}}}(x)} \tag{1.114}$$

and the pdf can be written as

$$f_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} \leq x) = \frac{\partial F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)/\partial y}{F_{\hat{\mathbf{x}}}(x)} = \frac{\int_{-\infty}^{x} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(q,y)\, dq}{\int_{-\infty}^{\infty}\int_{-\infty}^{x} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(q,y)\, dq\, dy}. \tag{1.115}$$

If we now take $B = \{x_1 < \hat{\mathbf{x}} \leq x_2\}$, we get

$$\begin{aligned} F_{\hat{\mathbf{y}}}(y|x_1 < \hat{\mathbf{x}} \leq x_2) &= \frac{P(x_1 < \hat{\mathbf{x}} \leq x_2, \hat{\mathbf{y}} \leq y)}{P(x_1 < \hat{\mathbf{x}} \leq x_2)} \\ &= \frac{F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x_2,y) - F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x_1,y)}{F_{\hat{\mathbf{x}}}(x_2) - F_{\hat{\mathbf{x}}}(x_1)} \end{aligned} \tag{1.116}$$

and the pdf

$$f_{\hat{\mathbf{y}}}(y|x_1 < \hat{\mathbf{x}} \leq x_2) = \frac{\int_{x_1}^{x_2} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x,y)\, dx}{\int_{x_1}^{x_2} f_{\hat{\mathbf{x}}}(x)\, dx}. \tag{1.117}$$

Let us consider, finally, the set $B = \{\hat{\mathbf{x}} = x\}$ as the limit $x_1 \to x_2$ of the previous case. Consequently, we define

$$F_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} = x) = \lim_{\Delta x \to 0} F_{\hat{\mathbf{y}}}(y|x < \hat{\mathbf{x}} \leq x + \Delta x). \tag{1.118}$$

From (1.116), we obtain

$$\begin{aligned} F_{\hat{\mathbf{y}}}(y \mid \hat{\mathbf{x}} = x) &= \lim_{\Delta x \to 0} \frac{F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x + \Delta x, y) - F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)}{F_{\hat{\mathbf{x}}}(x + \Delta x) - F_{\hat{\mathbf{x}}}(x)} \\ &= \frac{\partial F_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)/\partial x}{dF_{\hat{\mathbf{x}}}(x)/dx} \end{aligned} \tag{1.119}$$

which can be expressed as

$$F_{\hat{\mathbf{y}}}(y \mid \hat{\mathbf{x}} = x) = \frac{\int_{-\infty}^{y} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, p)\, dp}{f_{\hat{\mathbf{x}}}(x)}. \tag{1.120}$$

By taking the derivative with respect to x, we obtain the conditional pdf as

$$f_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} = x) = \frac{f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)}{f_{\hat{\mathbf{x}}}(x)} = \frac{f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)}{\int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)\, dy}. \tag{1.121}$$

Exchanging the role of x and y, we obtain

$$F_{\hat{\mathbf{x}}}(x \mid \hat{\mathbf{y}} = y) = \frac{\int_{-\infty}^{x} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(q, y)\, dq}{f_{\hat{\mathbf{y}}}(y)} \tag{1.122}$$

and

$$f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y) = \frac{f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)}{f_{\hat{\mathbf{y}}}(y)} = \frac{f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)}{\int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)\, dx}. \tag{1.123}$$

For the sake of simplicity, and if no confusion can arise, we will shorten the notation of the four last defined functions to

$$F_{\hat{\mathbf{y}}}(y|x),\ f_{\hat{\mathbf{y}}}(y|x),\ F_{\hat{\mathbf{x}}}(x|y),\ f_{\hat{\mathbf{x}}}(x|y). \tag{1.124}$$

Recall that, by Bayes theorem, if A and B are events and $B_1, B_2, \ldots$ is a partition of B, that is, $B = \cup_i B_i$ and $B_i \cap B_j = \emptyset$, then

$$P(B_i|A) = \frac{P(A|B_i)P(B_i)}{\sum_j P(A|B_j)P(B_j)}. \tag{1.125}$$

We now rephrase an equivalent of Bayes theorem in terms of pdfs. It follows from (1.121) and (1.123) that

$$f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y) = f_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} = x) f_{\hat{\mathbf{x}}}(x), \tag{1.126}$$

$$f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y) = f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y) f_{\hat{\mathbf{y}}}(y), \tag{1.127}$$

from which we obtain

$$f_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} = x) = \frac{f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y) f_{\hat{\mathbf{y}}}(y)}{f_{\hat{\mathbf{x}}}(x)}. \tag{1.128}$$

We now use (1.109) and (1.127) to derive

$$f_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y) f_{\hat{\mathbf{y}}}(y)\, dy \tag{1.129}$$

which when replaced in the denominator of (1.128) yields

$$f_{\hat{\mathbf{y}}}(y|\hat{\mathbf{x}} = x) = \frac{f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y) f_{\hat{\mathbf{y}}}(y)}{\int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y) f_{\hat{\mathbf{y}}}(y)\, dy} \tag{1.130}$$

which is a version of Bayes theorem in terms of pdfs.

In the application of these formulas in the following chapters, we will consider the case in which one of the random variables, say $\hat{\mathbf{y}}$, takes only discrete values. This means that its pdf takes the form

$$f_{\hat{\mathbf{y}}}(y) = \sum_i \text{Prob}(\hat{\mathbf{y}} = y_i)\delta(y - y_i), \tag{1.131}$$

and

$$f_{\hat{\mathbf{y}}}(y|x) = \sum_i \text{Prob}(\hat{\mathbf{y}} = y_i|x)\delta(y - y_i). \tag{1.132}$$

Replacing both expressions in

$$f_{\hat{\mathbf{y}}}(y) = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)\, dx = \int_{-\infty}^{\infty} f_{\hat{\mathbf{y}}}(y|x) f_{\hat{\mathbf{x}}}(x)\, dx \tag{1.133}$$

we derive

$$\text{Prob}(\hat{\mathbf{y}} = y_i) = \int_{-\infty}^{\infty} \text{Prob}(\hat{\mathbf{y}} = y_i|x) f_{\hat{\mathbf{x}}}(x)\, dx. \tag{1.134}$$

Also, replacing in (1.129), we obtain

$$f_{\hat{\mathbf{x}}}(x) = \sum_i f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{y}} = y_i)\text{Prob}(\hat{\mathbf{y}} = y_i). \tag{1.135}$$

Formulas that will be of interest in later chapters.

1.9 Markov Chains

As stated before, it is not difficult to generalize these concepts of joint probabilities to more than two random variables. For example, the pdf of n random variables $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n$ can be written in terms of conditional probabilities as

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_n) = f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x_2|x_1) f_{\hat{\mathbf{x}}_3}(x_3|x_1, x_2) \ldots f_{\hat{\mathbf{x}}_n}(x_n|x_1, \ldots, x_{n-1}). \tag{1.136}$$

This complicated expression adopts a much simpler form for a particular kind of random variables. A succession of random variables $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n$, is called a *Markov chain* if for any value of $m = 1, \ldots, n$ it fulfills

$$f_{\hat{\mathbf{x}}_m}(x_m|x_1, \ldots, x_{m-1}) = f_{\hat{\mathbf{x}}_m}(x_m|x_{m-1}). \tag{1.137}$$

That is, the pdf of $\hat{\mathbf{x}}_m$ conditioned to $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_{m-1}$ is equal to the pdf of $\hat{\mathbf{x}}_m$ conditioned only to $\hat{\mathbf{x}}_{m-1}$. From this property, (1.136) simplifies to

$$f_{\hat{\mathbf{x}}_1,\dots,\hat{\mathbf{x}}_N}(x_1, \dots, x_n) = f_{\hat{\mathbf{x}}_n}(x_n|x_{n-1}) f_{\hat{\mathbf{x}}_{n-1}}(x_{n-1}|x_{n-2}) \dots f_{\hat{\mathbf{x}}_2}(x_2|x_1) f_{\hat{\mathbf{x}}_1}(x_1). \tag{1.138}$$

Therefore, the joint pdf of $\hat{\mathbf{x}}_1, \dots, \hat{\mathbf{x}}_n$ is determined only by the knowledge of $f_{\hat{\mathbf{x}}_1}(x_1)$ and the conditional pdfs $f_{\hat{\mathbf{x}}_m}(x|y)$ (also known in this context as the transition pdf). We recall that $f_{\hat{\mathbf{x}}_m}(x|y)$ is actually a short way to write $f_{\hat{\mathbf{x}}_m}(x|\hat{\mathbf{x}}_{m-1} = y)$ with the meaning that $f_{\hat{\mathbf{x}}_m}(x|y)\,dx$ is the probability that the random variable $\hat{\mathbf{x}}_m$ adopts a value in the interval $(x, x + dx)$ provided that the random variable has taken the value $\hat{\mathbf{x}}_{m-1} = y$.

A Markov chain is called *homogeneous* if the transition probabilities $f_{\hat{\mathbf{x}}_m}(x|y)$ are independent of m. Thus, for a homogeneous Markov chain, we write the transition probabilities simply as $f(x|y)$.

It is easy to establish a relationship between $f_{\hat{\mathbf{x}}_{m+1}}(x)$ and $f_{\hat{\mathbf{x}}_m}(y)$ using the definition of conditional probability:

$$\begin{aligned} f_{\hat{\mathbf{x}}_{m+1}}(x) &= \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}_{m+1},\hat{\mathbf{x}}_m}(x, y)\,dy \\ &= \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}_{m+1}}(x|y) f_{\hat{\mathbf{x}}_m}(y)\,dy, \quad m \geq 1 \end{aligned} \tag{1.139}$$

which for a homogeneous chain reduces to

$$f_{\hat{\mathbf{x}}_{m+1}}(x) = \int_{-\infty}^{\infty} f(x|y) f_{\hat{\mathbf{x}}_m}(y)\,dy, \quad m \geq 1. \tag{1.140}$$

We can use this relation to *construct* the Markov chain. Starting from a given $f_{\hat{\mathbf{x}}_1}(x)$ initial pdf and a transition pdf $f(x|y)$, we can obtain the succession of random variables $\hat{\mathbf{x}}_m$, $m = 1, 2, \dots$ with the respective pdfs $f_{\hat{\mathbf{x}}_m}(x)$.

If the resulting pdfs $f_{\hat{\mathbf{x}}_m}(x)$ are all identical, $f_{\hat{\mathbf{x}}_m}(x) = f^{\text{st}}_{\hat{\mathbf{x}}}(x)$, $m = 1, 2, \dots$, we say that the Markov chain is stationary. For a stationary Markov chain, (1.140) becomes

$$f^{\text{st}}_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{\infty} f(x|y) f^{\text{st}}_{\hat{\mathbf{x}}}(y)\,dy. \tag{1.141}$$

It is not easy, in general, to solve the above integral equation to find the stationary pdf of a Markov chain with a given transition pdf $f(x|y)$. However, using

$$\int_{-\infty}^{\infty} f(y|x)\,dy = 1 \tag{1.142}$$

one can write (1.141) as

$$\int_{-\infty}^{\infty} f(y|x) f^{\text{st}}_{\hat{\mathbf{x}}}(x)\,dy = \int_{-\infty}^{\infty} f(x|y) f^{\text{st}}_{\hat{\mathbf{x}}}(y)\,dy \tag{1.143}$$

or, equivalently, as

$$\int_{-\infty}^{\infty} \left[f(y|x) f^{\text{st}}_{\hat{\mathbf{x}}}(x) - f(x|y) f^{\text{st}}_{\hat{\mathbf{x}}}(y) \right] dy = 0. \tag{1.144}$$

A way to satisfy this equation is by requiring the *detailed balance* condition

$$f(y|x) f^{\text{st}}_{\hat{\mathbf{x}}}(x) = f(x|y) f^{\text{st}}_{\hat{\mathbf{x}}}(y). \tag{1.145}$$

This is a simpler functional equation for $f^{\text{st}}_{\hat{\mathbf{x}}}(x)$ than the integral (1.141). Any solution $f^{\text{st}}_{\hat{\mathbf{x}}}(x)$ of the detailed balance condition[5] will satisfy (1.141), but the reverse is not always true.

Certainly, if a pdf $f^{\text{st}}_{\hat{\mathbf{x}}}(x)$ satisfies (1.141), then it is a stationary solution of the recurrence relation (1.140) such that $f_{\hat{\mathbf{x}}_m}(x) = f^{\text{st}}_{\hat{\mathbf{x}}}(x)$, $\forall m$, provided that $f_{\hat{\mathbf{x}}_1}(x) = f^{\text{st}}_{\hat{\mathbf{x}}}(x)$.

What happens when $f_{\hat{\mathbf{x}}_1}(x) \neq f^{\text{st}}_{\hat{\mathbf{x}}}(x)$? Will the recurrence (1.140) converge toward the stationary solution $f^{\text{st}}_{\hat{\mathbf{x}}}(x)$? A partial, but important, answer can be formulated as follows: If for every point x such that $f^{\text{st}}_{\hat{\mathbf{x}}}(x) > 0$ and for every initial condition $f_{\hat{\mathbf{x}}_1}(x)$, there exists a number m of iterations such that $f_{\hat{\mathbf{x}}_m}(x) > 0$ (*irreducibility* condition) and the recurrence relation (1.140) does not get trapped in cyclic loops, then $f^{\text{st}}_{\hat{\mathbf{x}}}(x)$ is the unique stationary solution and, furthermore, $\lim_{m\to\infty} f_{\hat{\mathbf{x}}_m}(x) = f^{\text{st}}_{\hat{\mathbf{x}}}(x)$. These conditions (irreducibility and noncyclic behavior) are summarized by saying that the Markov chain is *ergodic*. The irreducibility condition has a simple intuitive interpretation. It states that, independent of the initial condition, the recurrence relation (1.140) does not have ''forbidden'' zones, meaning that it is able to provide eventually a pdf with a nonzero probability to any point x such that $f^{\text{st}}_{\hat{\mathbf{x}}}(x) > 0$.

Finally, we can consider that the variable m of the Markov chain represents, in some suitable units, a ''time.'' In this sense, (1.140) introduces a dynamics in the space of pdfs. We will often make use of this dynamical interpretation of a Markov chain in the rest of the book. This dynamical interpretation, which can also be framed in the theory of Markov processes, can be made clear by introducing a ''time'' variable $t = m\delta t$, where δt is just the time unit. Then the function $f_{\hat{\mathbf{x}}_m}(x)$ becomes a two-variable function $f(x,t)$ and the evolution equation is given by

$$f_{\hat{\mathbf{x}}_{m+1}}(x) - f_{\hat{\mathbf{x}}_m}(x) = \int f(x|y) f_{\hat{\mathbf{x}}_m}(y)\, dy - f_{\hat{\mathbf{x}}_m}(x) \tag{1.146}$$

or using (1.142)

$$f_{\hat{\mathbf{x}}_{m+1}}(x) - f_{\hat{\mathbf{x}}_m}(x) = \int \left[f(x|y) f_{\hat{\mathbf{x}}_m}(y) - f(y|x) f_{\hat{\mathbf{x}}_m}(x) \right] dy. \tag{1.147}$$

With the interpretation $f_{\hat{\mathbf{x}}_m}(x) \to f(x,t)$, we can write it as

$$f_{\hat{\mathbf{x}}}(x, t+\delta t) - f_{\hat{\mathbf{x}}}(x,t) = \int \left[f(x|y) f_{\hat{\mathbf{x}}}(y,t) - f(y|x) f_{\hat{\mathbf{x}}}(x,t) \right] dy. \tag{1.148}$$

And, obviously, one is tempted to interpret the left-hand side as a partial derivative $\delta t \dfrac{\partial f(x,t)}{\partial t}$. We will explore this possibility in Chapter 8.

Further Reading and References

The material of this chapter is rather standard, and there are many books that cover it in more detail than our brief summary. The book by Papoulis and Pillai [1] is

5) Note that the detailed balance condition might have no solution, as we require that $f^{\text{st}}_{\hat{\mathbf{x}}}(x)$ is a pdf, that is, nonnegative and normalizable.

a good general introduction. The book by Grimmett and Stirzaker [2] is also of a more advanced level and is particularly interesting for the topic of Markov chains.

Exercises

1) A rod of length a is randomly dropped on a floor which has parallel lines drawn on it at a distance $\ell = 1$. Define what you think is meant by "randomly dropped" and, consequently, compute the probability that the rod intersects any of the lines (a problem due to Buffon).
2) A typist makes on average five mistakes in every 100 words. Find the probability that in a text of 1000 words the typist has made (a) exactly 10 mistakes and (b) at least 10 mistakes.
3) Use the Gaussian approximation to the Poisson distribution to find the probability that in a group of 10 000 people, at least 10 people were born on 1 January.
4) A study on the influence of the contraceptive pill on cervical cancer, published in the Lancet, **380**, 1740 (24 November 2007), analyzed a group of 52 082 women, 16 573 of which had taken the pill, and 35 509 have used other contraceptive methods. The study shows that the incidence of cervical cancer in the group of women that took the pill is 4.5 cases per 1000 women while in the group that did not take the pill it is 3.8 per 1000. Calculate the overall incidence rate in the group of 52 082 women. With this result, calculate the probability that a group of 16 573 women chosen randomly out of the 52 082 has a rate of 4.5 per 1000 or larger. Can we conclude that the pill increases the risk of cervical cancer, as was reported in some newspapers?
5) Prove that in the large k limit, the mean value and variance of the $\hat{\boldsymbol{\chi}}$ distribution, as given by (1.63) and (1.64) tend to $\langle \hat{\boldsymbol{\chi}} \rangle = \sqrt{k}$ and $\sigma^2[\hat{\boldsymbol{\chi}}] = 1/2$.
6) Prove that the correlation coefficient satisfies $|\rho[\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_j]| = 1$ if and only if there is a linear relationship $\hat{\mathbf{x}}_i = a\hat{\mathbf{x}}_j + b$ between the two random variables.
7) Compute the following integral:

$$L[J_1, J_2, \dots, J_N] = \sqrt{\frac{|A|}{(2\pi)^N}} \int_{\mathbb{R}^N} dx_1 \cdots dx_N \exp\left[-\frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{N} x_i A_{ij} x_j + \sum_{i=1}^{N} J_i x_i\right]$$

and use the result to prove Wick's theorem. Show, first, the equalities

$$\langle x_i \rangle = \left.\frac{\partial L}{\partial J_i}\right|_{\vec{J}=0}, \qquad \left\langle \prod_{k=1}^{n} x_{i_k} \right\rangle = \left. \frac{\partial^n L}{\prod_{k=1}^{n} \partial J_{i_k}} . \right|_{\vec{J}=0}$$

8) Prove that the sum of all symmetry factors in a Wick's diagram with n branches is $(n-1)!!$.
9) Use Wick's theorem to compute $\langle x_1^3 x_2^3 x_3 x_4 \rangle$, where the x_i's form a set a Gaussian variables of zero mean.

10) Let x be a Gaussian random variable of zero mean and variance σ^2, and let $g(x)$ be an arbitrary function. Prove Novikov's theorem:

$$\langle xg(x)\rangle = \sigma^2\langle g'(x)\rangle.$$

If $x = (x_1, \dots, x_N)$ is a set of jointly Gaussian random variables, generalize the theorem to

$$\langle x_i g(x)\rangle = \sum_{k=1}^{N}\langle x_i x_k\rangle \left\langle \frac{\partial g(x)}{\partial x_k}\right\rangle$$

11) Prove that, for a Gaussian variable x of mean μ and variance σ^2, it is $\langle e^{-x}\rangle = e^{-\mu+\sigma^2/2}$. For a general random variable, prove Jensen's inequality $\langle e^{-x}\rangle \geq e^{-\langle x\rangle}$.

12) Consider a homogeneous Markov chain with the following transition pdf:

$$f(x|y) = \frac{1}{\sigma\sqrt{2\pi(1-\lambda^2)}}\mathrm{e}^{-\frac{(x-\lambda y)^2}{2\sigma^2(1-\lambda^2)}}$$

with $|\lambda| < 1$. Prove that it is irreducible (hint: simply consider $f_{\hat{\mathbf{x}}_2}(x)$). Find its stationary pdf $f^{\mathrm{st}}_{\hat{\mathbf{x}}}(x)$ by searching for solutions of the detailed balance equation. Take as initial condition

$$f_{\hat{\mathbf{x}}_1}(x) = \frac{1}{a\sqrt{2\pi}}\mathrm{e}^{-\frac{x^2}{2a^2}},$$

compute $f_{\hat{\mathbf{x}}_n}(x)$, and prove that, effectively, $\lim_{n\to\infty} f_{\hat{\mathbf{x}}_n}(x) = f^{\mathrm{st}}_{\hat{\mathbf{x}}}(x)$.

13) Consider a homogeneous Markov chain with the following transition pdf:

$$f(x|y) = \begin{cases} \mathrm{e}^{-(x-y)}, & x \geq y, \\ 0, & x < y. \end{cases}$$

Prove that it is not irreducible. Furthermore, show that there are no solutions $f^{\mathrm{st}}_{\hat{\mathbf{x}}}(x)$ to the detailed balance condition (1.145) or the recurrence relation (1.140).

14) Consider a homogeneous Markov chain with the following transition pdf:

$$f(x|y) = \frac{b}{\pi}\frac{1}{1+b^2(x-\lambda y)^2}.$$

Show that it is irreducible. Prove that, if $|\lambda| < 1$, the stationary solution is

$$f^{\mathrm{st}}_{\hat{\mathbf{x}}}(x) = \frac{a}{\pi}\frac{1}{1+a^2x^2}$$

with $a = b(1-\lambda)$; prove also that there are no solutions $f^{\mathrm{st}}_{\hat{\mathbf{x}}}(x)$ to the detailed balance condition (1.145) if $\lambda \neq 0$.

15) Consider a homogeneous Markov chain with the following transition pdf:

$$f(x|y) = \begin{cases} \frac{2x}{y}, & x \in (0, y), \\ \frac{2(1-x)}{1-y}, & x \in (y, 1) \end{cases}$$

if $y \in (0, 1)$ and $f(x|y) = 0$ otherwise. Show that is irreducible, and find its stationary distribution.

2
Monte Carlo Integration

In this chapter, we review the basic algorithms for the calculation of integrals using random variables and define the general strategy based on the replacement of an integral by a sample mean.

2.1
Hit and Miss

The hit-and-miss method is the simplest of the integration methods that use ideas from probability theory. Although it is not very competitive in practical applications, it is very convenient in order to explain in a simple and comprehensive manner some of the ideas of more general methods. Let $y = g(x)$ be a real function which takes only bounded positive values in a finite interval $[a, b]$, that is, $0 \le g(x) \le c$. In Riemann's sense, the integral

$$I = \int_a^b g(x)\,dx \tag{2.1}$$

is nothing but the area of the region Ω between the X-axis and the curve given by $g(x)$. In the rectangle $S = \{(x, y),\ a \le x \le b,\ 0 \le y \le c\}$, we consider a pair of independent random variables $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ which are uniformly distributed in S, see Figure 2.1. The joint probability density function of $\hat{\mathbf{x}}$ and $\hat{\mathbf{y}}$ is

$$f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y) = \begin{cases} \dfrac{1}{c(b-a)} & (x, y) \in S, \\ 0 & (x, y) \notin S. \end{cases} \tag{2.2}$$

The probability that a point (x, y) distributed according to $f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)$ lies within Ω is, using (1.65)

$$p = \int_\Omega f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y)\,dx\,dy = \frac{1}{c(b-a)} \int_\Omega dx\,dy = \frac{I}{c(b-a)} \tag{2.3}$$

as $\int_\Omega dx\,dy$ is the area of Ω, or the integral I. The idea of the Monte Carlo integration is to read this equation as $I = c(b-a)p$. So, if we know the probability p, then we know the integral I. But how can we obtain the probability p that a point distributed according to a uniform distribution lies in region Ω? Simple: perform

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

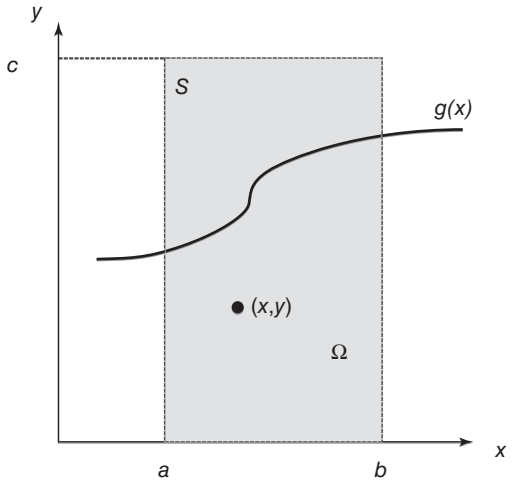


Figure 2.1 Schematics and definitions of the hit-and-miss method.

an experiment whose outcome is the pair (x, y), distributed according to (2.2), and compute from that experiment the probability p. It might help to imagine an archer who is sending arrows uniformly to region S. If he sends 100 arrows, and 78 of them are in region Ω, then an estimation of the probability is $p = 0.78$. The hit-and-miss method is a sophisticated manner of replacing the archer, giving us, hopefully, a precise value for p as well as the error in the calculation of the integral I.

Let us imagine, then, that we have an ideal (though very inefficient) archer who is able to send arrows that fall *uniformly* (and this is the key word) in points (x, y) of S. The point (x, y) can or cannot fall within Ω and, hence, the event "the point (x, y) is in Ω" can take the values "yes" or "no", a binary variable that follows a Bernoulli distribution. We make M independent repetitions of this Bernoulli experiment, generate the points $(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)$ uniformly distributed in S, and define the random variable $\hat{\mathbf{N}}_B$ as the number of points that belong to Ω, that is, those satisfying $y_i \leq g(x_i)$. We introduce a random variable $\hat{\mathbf{I}}_1$, defined from $\hat{p} = \dfrac{\hat{\mathbf{N}}_B}{M}$, as

$$\hat{\mathbf{I}}_1 = c(b-a)\frac{\hat{\mathbf{N}}_B}{M} = c(b-a)\hat{p}. \tag{2.4}$$

As $\hat{\mathbf{N}}_B$ follows a binomial distribution with mean value pM, it results that $\langle \hat{\mathbf{I}}_1 \rangle = I$. Such a random variable whose average value is equal to I is called an *unbiased estimator* of the integral. Using the results of the binomial distribution, we can compute also the standard deviation of this estimator as

$$\sigma[\hat{\mathbf{I}}_1] = \frac{c(b-a)}{M}\sigma[\hat{N}_B] = \frac{c(b-a)}{M}\sqrt{Mp(1-p)} = c(b-a)\sqrt{\frac{p(1-p)}{M}}. \tag{2.5}$$

We can now perform the experiment that consists in generating M points independently and uniformly distributed in the rectangle S and count how many times $\hat{\mathbf{N}}_B$ the points lie in Ω. This gives us a value for the random variable $\hat{p}$. Using the short-hand notation we derived from Chebyshev's theorem, we can write

$$I = \hat{\mathbf{I}}_1 \pm \hat{\sigma}[\hat{\mathbf{I}}_1] = c(b-a)\hat{p} \pm c(b-a)\sqrt{\frac{\hat{p}(1-\hat{p})}{M}} \tag{2.6}$$

with the usual confidence intervals for the error. If we make a large number of repetitions M, we can approximate the binomial distribution by a Gaussian distribution, and the confidence levels indicate that there is a 68% probability that the integral is indeed in the interval $(\hat{\mathbf{I}}_1 - \hat{\sigma}[\hat{\mathbf{I}}_1], \hat{\mathbf{I}}_1 + \hat{\sigma}[\hat{\mathbf{I}}_1])$, 95% that it is in the interval $(\hat{\mathbf{I}}_1 - 2\hat{\sigma}[\hat{\mathbf{I}}_1], \hat{\mathbf{I}}_1 + 2\hat{\sigma}[\hat{\mathbf{I}}_1])$, and so on. Consequently, we associate $\hat{\sigma}[\hat{\mathbf{I}}_1]$ to the error (in the statistical sense) of our estimator. The relative error is

$$\frac{\sigma[\hat{\mathbf{I}}_1]}{\langle \hat{\mathbf{I}}_1 \rangle} \approx \sqrt{\frac{1-p}{pM}}. \tag{2.7}$$

From this expression, we see (i) the relative error decreases as the inverse square root $M^{-1/2}$ of the number of repetitions M, and (ii) the error decreases with increasing p. Therefore, it is convenient to chose the, otherwise arbitrary, rectangle S as small as possible, which is equivalent to taking $c = \max(g(x))$ in the interval $x \in [a, b]$.

In a numerical algorithm, to replace our archer we need to generate random points (x, y) from a two-dimensional uniform distribution. This can be obtained by generating one random variable $\hat{\mathbf{x}}$ uniformly in the interval $[a, b]$ and another, independent, random variable $\hat{\mathbf{y}}$ uniformly in $(0, c)$. In practice, we use two independent random variables $\hat{\mathbf{u}}$, $\hat{\mathbf{v}}$ uniformly distributed in the interval $[0, 1]$ and the linear transformations $\hat{\mathbf{x}} = a + (b-a)\hat{\mathbf{u}}$, $\hat{\mathbf{y}} = c\hat{\mathbf{v}}$.

So let us assume that we have a way, a black box for the time being, an algorithm that enables us to generate independent $\hat{\mathbf{U}}(0, 1)$ random variables. We explain in Appendix A how to construct such an algorithm. It suffices to say that almost every scientific programming language or library incorporates a built-in function to do the job. The name of the function depends on the programming language and may even vary from compiler to compiler. Popular names are `rand()`, `random()`, `randu()`, and so on. Here, we will use here the generic name `ran_u()`. In the following, every time we see a call to the function `ran_u()`, the return is an independent $\hat{\mathbf{U}}(0, 1)$ random number uniformly distributed in the interval $(0, 1)$.

We implement the hit-and-miss method as a subroutine. The subroutine returns the estimator of the integral in the variable r and the error in s. In this and other examples, the important part of the code, the one that the reader should read in detail, is framed in a box.

```
subroutine mc1(g,a,b,c,M,r,s)
  implicit none
  double precision, intent (in) :: a,b,c
  integer, intent (in) :: M
  double precision, intent (out) :: r,s
```

```
  double precision :: g,p,u,v,ran_u
  integer :: nb,i
  external g
    nb=0
    do i=1,M
      u=ran_u()
      v=ran_u()
      if (g(a+(b-a)*u) > c*v) nb=nb+1
    enddo
    p=dble(nb)/M
    r=(b-a)*c*p
    s=sqrt(p*(1.d0-p)/M)*c*(b-a)
end subroutine mc1
```

For the sake of clarity, we include an example of a full program for calculating the area of the function $g(x) = x^2$ in the interval $[0, 1]$:

```
program area
  implicit none
  interface
    double precision function g(x)
      double precision, intent (in) :: x
    end function g
  end interface
  double precision :: a,b,c,r,s
  integer :: M
  a=0.d0
  b=1.d0
  c=1.d0
  M=100000
  call mc1(g,a,b,c,M,r,s)
  write (*,*) "The estimated value of the integral is", r
  write (*,*) "The estimated error is", s
end program area

double precision function g(x)
  implicit none
  double precision, intent (in) :: x
  g=x*x
end function g
```

The reader is encouraged to run this simple program and check that the result is compatible, within errors, with the exact value $I = 1/3$.

2.2 Uniform Sampling

Let us now explain a second more sophisticated method to perform integrals using random numbers and ideas from probability theory. We consider again the integral

$$I = \int_a^b g(x)\,dx \tag{2.8}$$

but now $g(x)$ does not need to be necessarily limited to take nonnegative values. We write this integral as

$$I = \int G(x) f_{\hat{\mathbf{x}}}(x)\,dx \tag{2.9}$$

with $G(x) = (b-a)g(x)$ and

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} \dfrac{1}{b-a} & a \le x \le b, \\ 0 & \text{otherwise.} \end{cases} \tag{2.10}$$

According to (1.21), this integral can be understood as the average value of the function $G(x)$ with respect to a random variable $\hat{\mathbf{x}}$ whose probability density function is $f_{\hat{\mathbf{x}}}(x)$, that is, a uniform random variable in the interval (a, b). If we now devise an experiment whose outcome is the random variable $\hat{\mathbf{x}}$ and repeat that experiment M times to obtain the values $x_1, \ldots, x_M$, we can compute from those values the sample mean and variance, as

$$\hat{\boldsymbol{\mu}}_M[G] = \frac{1}{M}\sum_{i=1}^{M} G(x_i), \tag{2.11}$$

$$\hat{\boldsymbol{\sigma}}_M^2[G] = \frac{1}{M}\sum_{i=1}^{M} G(x_i)^2 - \left(\frac{1}{M}\sum_{i=1}^{M} G(x_i)\right)^2. \tag{2.12}$$

Note that, contrary to (1.95), we have not included the term $\frac{M}{M-1}$ in the definition of $\hat{\boldsymbol{\sigma}}_M^2[G]$, as we will be using these formulas with a large value for M (on the order of thousands or millions) and it has only minimal numerical importance. Using (1.100), the integral can be estimated as

$$I = \hat{\boldsymbol{\mu}}_M[G] \pm \frac{\hat{\boldsymbol{\sigma}}_M[G]}{\sqrt{M}} \tag{2.13}$$

where the error has to be understood in the statistical sense of Chebyshev's theorem as explained in Section 1.6. If we write, for convenience, everything in terms of the original function $g(x)$, we obtain

$$I = (b-a)\hat{\boldsymbol{\mu}}_M[g] \pm (b-a)\frac{\hat{\boldsymbol{\sigma}}_M[g]}{\sqrt{M}} \tag{2.14}$$

where $\hat{\boldsymbol{\mu}}_M[g]$ and $\hat{\boldsymbol{\sigma}}_M[g]$ are the sample mean and standard deviation of the function $g(x)$.

It is very easy to implement this method because we already know how to generate values of a random variable $\hat{\mathbf{x}}$ which is uniformly distributed in an interval (a, b): take a $\hat{\mathbf{U}}(0, 1)$ variable $\hat{\mathbf{u}}$ and use the linear transformation $\hat{\mathbf{x}} = (b-a)\hat{\mathbf{u}} + a$. Let us give a simple computer program as follows:

```
subroutine mc2(g,a,b,M,r,s)
  implicit none
  double precision, intent (in) :: a,b
  integer, intent (in) :: M
  double precision, intent (out) :: r,s
  double precision :: g,g0,ran_u
  integer :: i
  external g

  r=0.d0
  s=0.d0
  do i=1,M
    g0=g(a+(b-a)*ran_u())
    r=r+g0
    s=s+g0*g0
  enddo
  r=r/M
  s=sqrt((s/M-r*r)/M)
  r=(b-a)*r
  s=(b-a)*s

end subroutine mc2
```

This simple algorithm is the uniform sampling method. If we think of the Riemann integral as the limiting sum of the function $g(x)$ over an infinite number of points in the interval (a, b), the uniform sampling method replaces that infinite sum by a finite sum (2.11) over points randomly distributed in the same interval.

2.3 General Sampling Methods

Similar ideas can be used for the integral

$$I = \int G(x) f_{\hat{\mathbf{x}}}(x)\, dx \tag{2.15}$$

where, again, there are no restrictions about the function $G(x)$, but we demand that $f_{\hat{\mathbf{x}}}(x)$ has the required properties of a probability density function: that is, nonnegative and normalized. The key point is to understand the integral as the average value of the function $G(x)$ with respect to a random variable $\hat{\mathbf{x}}$ whose probability density function is $f_{\hat{\mathbf{x}}}(x)$:

$$I = E[G]. \tag{2.16}$$

If we devise now an experiment whose outcome is the random variable $\hat{\mathbf{x}}$ with probability density function $f_{\hat{\mathbf{x}}}(x)$ and repeat that experiment M times to obtain the values $x_1, \ldots, x_M$, we can compute from those values the sample mean and variance using (2.11) and (2.12). Then, as before, the integral can be approximated by (2.13). This simple but deep result is the basis of the general sampling method: to evaluate integral (2.15), we consider it as the average of $G(x)$ with respect

to a random variable $\hat{\mathbf{x}}$ whose probability distribution function is $f_{\hat{\mathbf{x}}}(x)$ and use the approximation given by the sample mean and the error given by the sample variance, formulas (2.11)–(2.13), but using for $x_i, i = 1, \ldots, M$ values of the random variable $\hat{\mathbf{x}}$ whose pdf is $f_{\hat{\mathbf{x}}}(x)$. It is straightforward to write a computer program to implement this algorithm.

```
subroutine mc3(g,ran_f,M,r,s)
  implicit none
  integer, intent (in) :: M
  double precision, intent (out) :: r,s
  double precision :: g,ran_f,g0
  integer :: i
  external g,ran_f

    r=0.d0
    s=0.d0
    do i=1,M
      g0=g(ran_f())
      r=r+g0
      s=s+g0*g0
    enddo
    r=r/M
    s=sqrt((s/M-r*r)/M)

end subroutine mc3
```

The main difference with respect to the uniform sampling is the substitution of the values of the uniform distribution `(b-a)ran_u()+a` distributed according to a uniform distribution in (a, b) by the values `ran_f()` distributed according to the distribution $f_{\hat{\mathbf{x}}}(x)$. And how do we implement this function? Although we will devote next chapter entirely to this question, let us give now some basic results.

2.4 Generation of Nonuniform Random Numbers: Basic Concepts

The basic method to generate values of a random variable $\hat{\mathbf{x}}$ distributed according to the probability distribution function $f_{\hat{\mathbf{x}}}(x)$ uses the theorem proven in Section 1.3.4:

Theorem. If $\hat{\mathbf{x}}$ is a random variable whose cumulative probability function is $F_{\hat{\mathbf{x}}}(x)$, then the change of variables $\hat{\mathbf{u}} = F_{\hat{\mathbf{x}}}(\hat{\mathbf{x}})$ yields a random variable $\hat{\mathbf{u}}$ uniformly distributed in the interval $(0, 1)$, i.e., a $\hat{\mathbf{U}}(0, 1)$ variable.

We read this theorem backward: if $\hat{\mathbf{u}}$ is a $\hat{\mathbf{U}}(0, 1)$ random variable, then the probability distribution function of $\hat{\mathbf{x}} = F^{-1}_{\hat{\mathbf{x}}}(\hat{\mathbf{u}})$ is $f_{\hat{\mathbf{x}}}(x)$, see Figure 2.2. Let us see an example. We want to generate random numbers distributed according to an

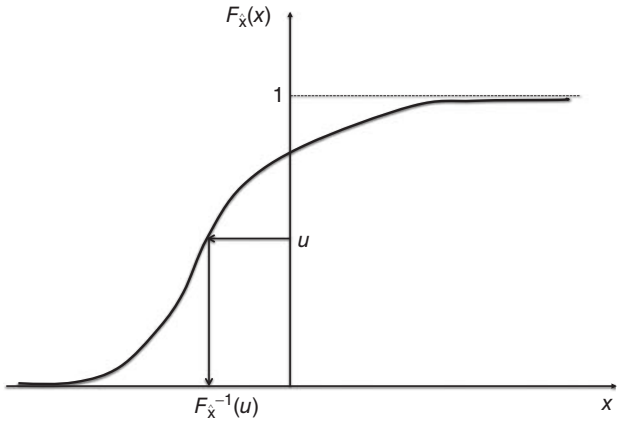


Figure 2.2 Inversion of the cdf $F_{\hat{\mathbf{x}}}(x)$. If u is the value of a random variable $\hat{\mathbf{u}}$ uniformly distributed in the $(0,1)$ interval, then $x = F_{\hat{\mathbf{x}}}^{-1}(u)$ follows the corresponding pdf $f_x(x) = dF_x(x)/dx$.

exponential distribution

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ a\exp(-ax), & x \geq 0. \end{cases} \tag{2.17}$$

We first compute the cumulative distribution

$$F_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{x} f_{\hat{\mathbf{x}}}(x')\,dx' = 1 - \exp(-ax), x \geq 0, \tag{2.18}$$

and then invert $x = F_{\hat{\mathbf{x}}}^{-1}(u)$ by solving $u = F_{\hat{\mathbf{x}}}(x)$ or $x = \frac{-1}{a}\log(1-u)$. This means that, to generate random numbers x distributed according to an exponential distribution, we need to generate numbers u uniformly distributed in the $(0,1)$ and then apply $x = \frac{-1}{a}\log(1-u)$. As $1 - \hat{\mathbf{u}}$ is obviously also a $\hat{\mathbf{U}}(0,1)$ variable, we can simply use the formula

$$x = \frac{-1}{a}\log(u). \tag{2.19}$$

The algorithm can be implemented in the following simple program[1]:

```
double precision function ran_exp(a)
  implicit none
  double precision, intent (in) :: a
  double precision :: ran_u

    ran_exp=-log(ran_u())/a

end function ran_exp
```

1) A possible problem with this algorithm one needs to be aware of is that it might occur that the uniform random number u takes exactly the value $u = 0$. This can be avoided by generating the uniform numbers in a suitable way as discussed in Appendix A.

Let us see some more examples.

Cauchy Distribution: The probability distribution function is

$$f_{\hat{\mathbf{x}}}(x) = \frac{1}{\pi}\frac{1}{1+x^2}. \tag{2.20}$$

The cumulative function is

$$F_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{x} f_{\hat{\mathbf{x}}}(x)\,dx = \int_{-\infty}^{x} \frac{1}{\pi}\frac{1}{1+x^2} = \frac{1}{2} + \frac{1}{\pi}\arctan(x) \tag{2.21}$$

and its inverse function is

$$x = \tan\left(\pi\left(u - \frac{1}{2}\right)\right). \tag{2.22}$$

To generate random values distributed according to the more general distribution

$$f_{\hat{\mathbf{x}}}(x) = \frac{a}{\pi}\frac{1}{1+a^2(x-x_0)^2} \tag{2.23}$$

we do not need to repeat the whole process. We simply note that the variable $\hat{\mathbf{z}}$ defined by $\hat{\mathbf{z}} = a(\hat{\mathbf{x}} - x_0)$ follows the Cauchy distribution $f_{\hat{\mathbf{z}}}(z) = \frac{1}{\pi}\frac{1}{1+z^2}$ and use $x = x_0 + a^{-1}\tan(\pi(u - \frac{1}{2}))$.

A Power-Law Distribution in a Bounded Domain: Consider the random variable $\hat{\mathbf{x}}$ distributed according to

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} (1+a)x^a, & x \in [0,1], \\ 0, & x \notin [0,1] \end{cases} \tag{2.24}$$

with $a > -1$ (otherwise, it is not normalizable). The cumulative distribution is $F_{\hat{\mathbf{x}}}(x) = x^{a+1}$ if $x \in [0,1]$ and the inversion is $x = u^{1/(1+a)}$.

A Power-Law Distribution in an Infinite Domain: This kind of random variables appears in many modern applications of the field of complex systems. Roughly speaking, a random variable defined in the infinite domain $(0,\infty)$ is said to have a power-law distribution $f_{\hat{\mathbf{x}}}(x)$ with exponent $a > 0$ if $f_{\hat{\mathbf{x}}}(x) \sim x^{-a}$ for large x. The problem is that this definition is not very precise. There are many functions $f_{\hat{\mathbf{x}}}(x)$ that behave as $f_{\hat{\mathbf{x}}}(x) \sim x^{-a}$ for large x, and one needs to specify which one is needed in a particular problem. Note that one cannot use $f_{\hat{\mathbf{x}}}(x) = Cx^{-a}$ for all $x \in (0,\infty)$, as this function is not normalizable for any value of the exponent a. Many functions $f_{\hat{\mathbf{x}}}(x)$ have been proposed. We will consider here the specific example

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ \frac{a-1}{x_0}\left(1 + \frac{x}{x_0}\right)^{-a}, & x \in [0,\infty) \end{cases} \tag{2.25}$$

which is well defined if $a > 1$ and $x_0 > 0$. The cumulative function for $x \geq 0$ is

$$F_{\hat{\mathbf{x}}}(x) = 1 - \left(1 + \frac{x}{x_0}\right)^{1-a} \tag{2.26}$$

and the inversion $x = F^{-1}_{\hat{\mathbf{x}}}(u)$ leads to

$$x = (u^{\frac{1}{1-a}} - 1)\, x_0 \tag{2.27}$$

where we have replaced u by $1 - u$ as they are statistically equivalent.

Other possibilities for a *bona fide* power-law distribution imply the use of cutoffs. For example, a distribution with a cutoff at small values of x

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < x_0, \\ \frac{a-1}{x_0} \left(\frac{x}{x_0}\right)^{-a}, & x \in [x_0, \infty), \end{cases} \tag{2.28}$$

is valid for $x_0 > 0$ and $a > 1$. It is easy to show that values of this random variable can be generated using $x = u^{1/(1-a)}\, x_0$.

Rayleigh Distribution: A random variable $\hat{\mathbf{r}}$ that follows the Rayleigh distribution has as probability density function

$$f_{\hat{\mathbf{r}}}(r) = \begin{cases} 0, & r < 0, \\ re^{-\frac{1}{2}r^2}, & 0 \le r \le \infty. \end{cases} \tag{2.29}$$

The cumulative function is

$$F_{\hat{\mathbf{r}}}(r) = \int_{-\infty}^{r} f_{\hat{\mathbf{r}}}(r)\, dr = 1 - e^{-\frac{1}{2}r^2} \tag{2.30}$$

and the inverse function

$$r = \sqrt{-2\log(1-u)} \equiv \sqrt{-2\log(u)} \tag{2.31}$$

where, again, we have replaced u by $1 - u$ as they are statistically equivalent. We will be using this formula later when discussing the Box–Muller–Wiener algorithm for the Gaussian distribution.

Gaussian Distribution. If $\hat{\mathbf{x}}$ is a Gaussian random variable of mean μ and variance σ^2, with probability density function

$$f_{\hat{\mathbf{x}}}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2.32}$$

then $\hat{\mathbf{z}} = \frac{\hat{\mathbf{x}}-\mu}{\sigma}$ is a Gaussian random variable of zero mean and variance 1 with probability density function

$$f_{\hat{\mathbf{z}}}(z) = \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}. \tag{2.33}$$

Then, in order to generate $\hat{\mathbf{x}}$, all we need to do is generate $\hat{\mathbf{z}}$ and apply the linear change $\hat{\mathbf{x}} = \sigma\hat{\mathbf{z}} + \mu$. The cumulative function of $\hat{\mathbf{z}}$ is

$$F_{\hat{\mathbf{z}}}(z) = \int_{-\infty}^{z} f_{\hat{\mathbf{z}}}(z)\, dz = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} e^{-\frac{z^2}{2}}\, dz = \frac{1}{2}\left(1 + \operatorname{erf}\left(z/\sqrt{2}\right)\right) \tag{2.34}$$

and values of the random variable $\hat{\mathbf{z}}$ are obtained from

$$z = \sqrt{2}\, \operatorname{erf}^{-1}(2u - 1). \tag{2.35}$$

A possible problem with this expression is that the inverse error function $\mathrm{erf}^{-1}(x)$ is not standard in most programming languages and you will need to search for it in an appropriate scientific library. For instance, in Intel's Math Kernel Library it is called `erfinv(x)`. One could use, alternatively, some of the good approximations to the inverse error function $z = F_{\hat{z}}^{-1}(u)$,

$$z \approx t - \frac{c_0 + c_1 t + c_2 t^2}{1 + d_1 t + d_2 t^2 + d_3 t^3}, \qquad t = \sqrt{-2\log(1-u)} \tag{2.36}$$

and the numerical values c_0=2.515517, c_1=0.802853, c_2=0.010328, $d1$=1.432788, d_2=0.189269, and d_3=0.001308. The error is less than 4.5×10^{-4} if $0.5 \leq u \leq 1.0$. For $0 < u < 0.5$, we can use the symmetry property of the Gaussian distribution around $x = 0$ to write up the following computer program:

```
double precision function ran_g()
  implicit none
  double precision, parameter :: c0=2.515517d0
  double precision, parameter :: c1=0.802853d0
  double precision, parameter :: c2=0.010328d0
  double precision, parameter :: d1=1.432788d0
  double precision, parameter :: d2=0.189269d0
  double precision, parameter :: d3=0.001308d0
  double precision :: u,t,ran_u

 u=ran_u()
  if (u > 0.5d0) then
    t=sqrt(-2.d0*log(1.0d0-u))
    ran_g=t-(c0+t*(c1+c2*t))/(1.d0+t*(d1+t*(d2+t*d3)))
  else
    t=sqrt(-2.d0*log(u))
    ran_g=-t+(c0+t*(c1+c2*t))/(1.d0+t*(d1+t*(d2+t*d3)))
  endif

end function ran_g
```

Besides the intrinsic error, which might or might not be important in a particular application, this approximation to the inverse error function is relatively slow as it involves the calculation of a square root, a logarithm, and a ratio of polynomials. In the next chapter, we will develop alternative algorithms for the Gaussian distribution that use faster functions.

The problem of the lack of a good algorithm for the calculation of the inverse cumulative distribution function is very general and, sometimes, it is a bottleneck for the implementation of the method described in this section. For example, consider a variable distributed according to the distribution

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ xe^{-x}, & x \geq 0. \end{cases} \tag{2.37}$$

(It belongs to the Gamma distribution family; concretely, it is the $\hat{\mathbf{\Gamma}}(2,1)$ distribution, see more in the next chapter.) The cumulative distribution is

$$F_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ 1 - (1+x)e^{-x}, & x > 0. \end{cases} \tag{2.38}$$

Now, given a value u uniformly distributed in the interval $(0, 1)$, all we need to do is to find the positive solution x of the nonalgebraic equation $u = F_{\hat{\mathbf{x}}}(x) = 1 - (1+x)e^{-x}$. This can be solved, for example, using Newton–Raphson method with the recurrence relation[2)]

$$x_{n+1} = x_n - \frac{F_{\hat{\mathbf{x}}}(x_n) - u}{f_{\hat{\mathbf{x}}}(x_n)} \tag{2.39}$$

where we have replaced $F'_{\hat{\mathbf{x}}}(x) = f_{\hat{\mathbf{x}}}(x)$. As an initial condition, we can expand $F_{\hat{\mathbf{x}}}(x) = x^2/2 + O(x^3) = u$ and use $x_0 = \sqrt{2u}$. A possible program could be as follows:

```
double precision function ran_gamma2()
  implicit none
  double precision :: u,x,xn,fx,fxc,ran_u

   fx(x)=x*exp(-x)
   fxc(x)=1.d0-exp(-x)*(1.d0+x)
   u=ran_u()
   x=sqrt(2.d0*u)
   xn=x-(fxc(x)-u)/fx(x)
   do while(abs(xn-x) > 1.d-8)
     xn=x
     x=x-(fxc(x)-u)/fx(x)
   enddo
   ran_gamma2=x

end function ran_gamma2
```

The same procedure can be used, for instance, to generate the distribution (the $\hat{\mathbf{\Gamma}}(3, 1)$ member of the gamma family)

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ \frac{1}{2}x^2 e^{-x}, & x > 0 \end{cases} \tag{2.40}$$

with a cumulative distribution

$$F_{\hat{\mathbf{x}}}(x) = \begin{cases} 0, & x < 0, \\ 1 - \left(1 + x + \frac{1}{2}x^2\right)e^{-x}, & x > 0. \end{cases} \tag{2.41}$$

The Newton–Raphson algorithm can be implemented in the following program using the initial value $x_0 = (6u)^{1/3}$ obtained by expanding $F_{\hat{\mathbf{x}}}(x) = \frac{1}{6}x^3 + O(x^4)$:

2) Other methods of solution will also work. For instance, write it in the form $x = -\log\left(\frac{u}{1+x}\right)$, replacing u by $1 - u$, and use the recurrence relation $x_{n+1} = -\log\left(\frac{u}{1+x_n}\right)$ with $x_0 = 1$. The reader can check that this procedure always converges to the appropriate positive solution for x.

```
double precision function ran_gamma3()
  implicit none
  double precision :: u,x,xn,fx,fxc,ran_u
```

```
  fx(x)=0.5d0*x*x*exp(-x)
  fxc(x)=1.d0-exp(-x)*(1.d0+x*(1.d0+0.5d0*x))
  u=ran_u()
  x=(6.d0*u)**(1.d0/3.d0)
  xn=x-(fxc(x)-u)/fx(x)
  do while(abs(xn-x) > 1.d-8)
    xn=x
    x=x-(fxc(x)-u)/fx(x)
  enddo
  ran_gamma3=x
```

```
end function ran_gamma3
```

In the next chapter, however, we will see more efficient methods to generate values of random variables distributed according to Gamma-type distributions.

The same method of the inversion of the cumulative distribution function can be used in the case of discrete random variables, those taking values from a numerable (maybe infinite) set $(x_1, x_2, \ldots)$ ordered as $x_1 < x_2 < x_3 \ldots$. If p_i is the probability that the random variable $\hat{\mathbf{x}}$ takes the value x_i, the pdf is

$$f_{\hat{\mathbf{x}}}(x) = \sum_{i=1} p_i \delta(x - x_i) \tag{2.42}$$

and the corresponding cdf is

$$F_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{x} f_{\hat{\mathbf{x}}}(x)\, dx = \sum_{i=1}^{m} p_i. \tag{2.43}$$

Here, m stands for the largest integer number for which $x_m \leq x$. The cdf is a step function. To invert $F_{\hat{\mathbf{x}}}(x)$, we need to determine to which interval $[x_m, x_{m+1})$ does $F^{-1}_{\hat{\mathbf{x}}}(u)$ belong, see Figure 2.3. In general, it is not easy to find the interval $[x_m, x_{m+1})$, so it is better to consider the equivalent problem of finding the maximum value of m for which

$$F_{\hat{\mathbf{x}}}(x_m) = \sum_{i=1}^{m} p_i \leq u. \tag{2.44}$$

If the number of values that the random variable $\hat{\mathbf{x}}$ can take is not too large, it might be convenient to check directly to which interval the uniform random number belongs. For instance, let us consider the distribution

$$f_{\hat{\mathbf{x}}}(x) = \frac{1}{8}\delta(x) + \frac{3}{8}\delta\left(x - \frac{1}{2}\right) + \frac{1}{6}\delta(x-2) + \frac{1}{3}\delta(x-4) \tag{2.45}$$

for which we can use the following program:

```
double precision function ran_discrete_f()
  implicit none
  double precision :: u,s1,s2,s3,ran_u
```

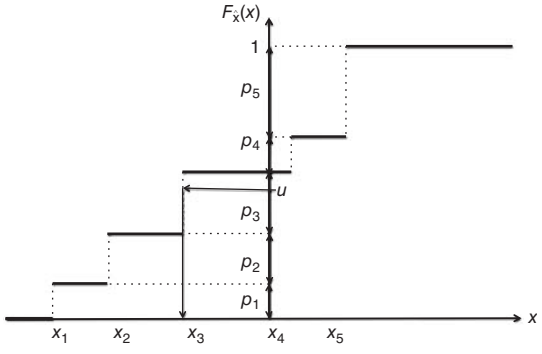


Figure 2.3 Inversion of the cdf for a discrete distribution.

```
  u=ran_u()
  s1=1.d0/8.d0
  s2=s1+3.d0/8.d0
  s3=s2+1.d0/6.d0
  if (u < s1) then
    ran_f=0.d0
  elseif(u < s2) then
    ran_f=0.5d0
  elseif(u < s3) then
    ran_f=2.d0
  else
    ran_f=4.d0
  endif
```

```
end function ran_discrete_f
```

A particular case is that of the Bernoulli distribution for which the cumulative distribution function is (1.29). The inverse function takes only two values, namely

$$F_{\hat{x}}^{-1}(u) = \begin{cases} 0 & u < 1-p, \\ 1 & u \geq 1-p. \end{cases} \tag{2.46}$$

See Figure 1.2. Hence, a program to generate a Bernoulli random variable with parameter p could be as follows:

```
integer function iran_bern(p)
  implicit none
  double precision, intent (in) :: p
  double precision :: ran_u
```

```
  if (ran_u() < 1.0d0-p) then
    iran_bern=0
  else
    iran_bern=1
  endif
```

```
end function iran_bern
```

Replacing u by $1-u$ (as they are statistically equivalent), the relevant lines of this program can also be written as follows:

```
if (ran_u() > p) then
  iran_bern=0
else
  iran_bern=1
endif
```

If the random variable $\hat{\mathbf{x}}$ takes N values $x_1, \ldots, x_N$ with the same probability, that is, $p_i = \frac{1}{N}$, it is possible to find the desired interval $[x_m, x_{m+1})$ as[3] $m = [Nu] + 1$. Hence, to sample

$$f_{\hat{\mathbf{x}}}(x) = \sum_{i=1}^{N} \frac{1}{N} \delta(x - x_i) \tag{2.47}$$

we draw u from a $\hat{\mathbf{U}}(0,1)$ distribution and take[4] $x = x_{[Nu]+1}$.

If the discrete random variable can take an infinitely numerable number of values, it is usually difficult to invert efficiently the cdf $F_{\hat{\mathbf{x}}}(x)$. A notable exception is the **geometric distribution**, which takes integer values $x = i$ with probability $p_i = pq^i$ (with $q = 1 - p$). The cdf is

$$F_{\hat{\mathbf{x}}}(m) = \sum_{i=0}^{m} p_i = \sum_{i=0}^{m} pq^i = 1 - q^{m+1}. \tag{2.48}$$

We need to find the maximum integer value m for which

$$1 - q^{m+1} \leq u \rightarrow q^{m+1} \geq 1 - u \equiv u \tag{2.49}$$

or

$$m = \left[\frac{\log(u)}{\log(q)} \right]. \tag{2.50}$$

The program could look as follows[5]:

```
integer function iran_geo(p)
  implicit none
  double precision, intent (in) :: p
  double precision :: ran_u
```

```
  iran_geo=int(log(ran_u())/log(1.d0-p))
```

```
end function iran_geo
```

3) Recall that $[z]$ denotes the integer part of the real variable z.
4) This algorithm assumes that the value $u = 1$ can never appear exactly. This is true for the random numbers of the form $u = m/M$ with $0 \leq m \leq M - 1$ discussed in Appendix A.
5) Now we need to make sure that the random number generator cannot return exactly the value $u = 0$.

In other cases of discrete variables, the cdf $F_{\hat{\mathbf{x}}}(x)$ might be difficult to invert. Consider the discrete distribution $p_i = (1+i)p^2q^i$, with $q = 1-p$, which is a modified geometric distribution. The cumulative distribution is

$$F_{\hat{\mathbf{x}}}(m) = \sum_{i=0}^{m} p_i = 1 - q^{m+1}(1 + p(m+1)) \tag{2.51}$$

and we need to find the maximum integer value m for which $F_{\hat{\mathbf{x}}}(m) \leq u$, or

$$q^{m+1}(1 + p(m+1)) \geq 1 - u \equiv u. \tag{2.52}$$

This equation needs to be solved numerically. For instance, we can set $x = m+1$ and set the iteration scheme

$$\begin{aligned} x_0 &= \frac{\log(u)}{\log(q)}, \\ x_{n+1} &= \frac{\log u - \log(1 + px_n)}{\log q} \end{aligned} \tag{2.53}$$

and iterate until $|x_n - x_{n+1}|$ is less than, say, 10^{-6}. Next, we set $m = [x]$, the integer part of x. The program would be as follows:

```
integer function iran_modgeo(p)
  implicit none
  double precision, intent (in) :: p
  double precision :: a,v,x0,x,ran_u
  a=log(1.d0-p)
  v=log(ran_u())
  x0=v/a
  x=x+1.d0
  do while(abs(x0-x) > 1.d-6)
    x=x0
    x0=(v-dlog(1.d0+p*x))/a
  enddo
  iran_modgeo=int(x)
end function iran_modgeo
```

Sometimes, it is not even possible to find a suitable analytical expression for $F_{\hat{\mathbf{x}}}(m) = \sum_{i=0}^{m} p_i$. In this case, what we can do is to find directly the solution of $F_{\hat{\mathbf{x}}}(m) \leq u$. This can be generically implemented as follows:

```
integer function iran_generic(p)
  implicit none
  double precision :: p
  double precision :: F,v,ran_u
  integer :: i
  external p
```

```
  v=ran_u()
  F=p(0)
  i=0
  do while(F < v)
    i=i+1
    F=F+p(i)
  enddo
  iran_generic=i
```

```
end function iran_generic
```

where p(i) is a external function that returns the value of p_i. For example, for the distribution $p_i = \frac{6}{\pi^2}(i+1)^{-2}$, the function p(i) can be written as follows:

```
double precision function p(i)
  implicit none
  integer, intent (in) :: i
  double precision, parameter :: coeff=0.607927101854026626 1d0
  p=coeff/(1+i)**2
end function p
```

since, in this case, $F_{\hat{\mathbf{x}}}(m) \leq u$ corresponds to $\sum_{i=0}^{m}(i+1)^{-2} \leq \frac{\pi^2}{6}u$. Therefore, one can also implement directly the following:

```
integer function iran_pot2()
  implicit none
  double precision, parameter :: pi26=1.644934066848226d0
  double precision :: v,F,ran_u
  integer :: m
```

```
  v=pi26*ran_u()
  F=1.d0
  m=0
  do while(F < v)
    m=m+1
    F=F+1.d0/(m+1)**2
  enddo
  iran_pot2=m
```

```
end function iran_pot2
```

The same ideas can be applied to the Poisson distribution of parameter λ, $\hat{\mathbf{P}}(\lambda)$, which takes the integer values $i = 0, 1, 2, \ldots$ with probability $p_i = e^{-\lambda}\lambda^i/i!$. Given a number u extracted from a uniform $\hat{\mathbf{U}}(0,1)$ distribution, we need to find the maximum value m that satisfies

$$\sum_{i=0}^{m} e^{-\lambda}\frac{\lambda^i}{i!} \leq u. \tag{2.54}$$

Although this sum can be written out in terms of known functions, namely

$$\sum_{i=0}^{m} e^{-\lambda}\frac{\lambda^i}{i!} = \frac{\Gamma(m+1,\lambda)}{\Gamma(m+1)} \tag{2.55}$$

with $\Gamma(x,\lambda) = \int_\lambda^\infty dt\, t^{x-1}e^{-t}$ being the incomplete Gamma function, this does not help us in the solution of (2.54), as both $\Gamma(x)$ and $\Gamma(x,\lambda)$ are not part of the standard set of functions of most compiling languages. A better alternative is to solve (2.54) directly. We first write it as

$$\sum_{i=0}^{m} \frac{\lambda^i}{i!} \leq ue^{\lambda}. \tag{2.56}$$

We find the value of m by adding terms to the left-hand side of this expression until the inequality is not satisfied. In the calculation of the terms of the sum, it is helpful to use the relation

$$\frac{\lambda^{i+1}}{(i+1)!} = \frac{\lambda}{i+1}\frac{\lambda^i}{i!} \tag{2.57}$$

leading to the following algorithm:

```
integer function iran_poisson(lambda)
  implicit none
  double precision, intent (in) :: lambda
  double precision :: v,F,a,ran_u
  integer :: m
```

```
  F=1.d0
  v=dexp(lambda)*ran_u()
  a=F
  m=0
  do while(F < v)
    a=a*lambda/(m+1)
    F=F+a
    m=m+1
  enddo
  iran_poisson=m
```

```
end function iran_poisson
```

This algorithm works fine for small values of λ but its efficiency worsens for large λ. The reason is simple to understand. We know that the Poisson distribution has mean value and variance equal to λ. This means that most of the times the generated values belong to the interval $\lambda - \sqrt{\lambda}, \lambda + \sqrt{\lambda}$. As we begin searching from $m = 0$, it takes, on average, λ sums to solve (2.54). Why not then start by searching the solution of (2.54) using a value of m close to λ, namely $m = [\lambda]$, instead of starting from $m = 0$? All we need to do is to compute first the partial sum

$$F = \sum_{i=0}^{[\lambda]} \frac{\lambda^i}{i!} \tag{2.58}$$

as well as the latest term $a = \frac{\lambda^{[\lambda]}}{[\lambda]!}$. If $F < ue^{\lambda}$, this means that the required value of m that satisfies (2.56) is larger than $[\lambda]$, and we keep on increasing m until we find its largest value for which (2.56) holds. If, on the other hand, $F > ue^{\lambda}$, it is $m < [\lambda]$, and we decrease m and subtract terms from the sum until the condition (2.56) is

satisfied. In this way, we need to sum the order of $\sqrt{\lambda}$ terms to find the solution of (2.54). This is a big improvement if λ is large.

As a practical issue, in this case, to find the corresponding terms of the sum we use

$$\frac{\lambda^{i-1}}{(i-1)!} = \frac{i}{\lambda}\frac{\lambda^i}{i!}. \tag{2.59}$$

Here is a program implementing these ideas:

```
integer function iran_poisson2(lambda,a0,F0)
  implicit none
  double precision, intent (in) :: lambda
  double precision :: v,F,F0,a,a0,ran_u
  integer :: m
```

```
  v=dexp(lambda)*ran_u()
  m=lambda
  a=a0
  F=F0
  if (F < v) then
    do  while(F < v)
      m=m+1
      a=a*lambda/m
      F=F+a
    enddo
  else
    do  while(F > v)
      m=m-1
      F=F-a
      a=a*(m+1)/lambda
    enddo
    m=m+1
  endif
  iran_poisson2=m
```

```
end function iran_poisson2
```

where, in the main program, we need to add, before the first call to the function `iran_poisson2()`, the lines

```
  F0=1.0d0
  a0=F0
  do i=1,lambda
    a0=a0*lambda/i
    F0=F0+a0
  enddo
```

In any event, neither of these programs is useful for very large λ, as the calculation of e^{λ} can easily result in an overflow. There are alternative ways to generate a Poisson distribution. We will review them in a later chapter.

The final example is that of the binomial distribution, where $p_i = \binom{N}{i} p^i (1-p)^{N-i}$. According to the general procedure, we need to find the maximum value of m for

which

$$\sum_{i=0}^{m} \binom{N}{i} p^i (1-p)^{N-i} \leq u \tag{2.60}$$

where u is a random number taken from a $\hat{\mathbf{U}}(0,1)$ distribution. When programming this algorithm, it is useful to use the relation between two consecutive terms in the sum

$$\binom{N}{i+1} p^{i+1} (1-p)^{N-i-1} = \frac{p}{1-p} \frac{N-i}{i+1} \binom{N}{i} p^i (1-p)^{N-i} \tag{2.61}$$

which is valid for $i = 0, 1, \ldots, N-1$, and that the first term is $(1-p)^N$. A possible implementation is as follows:

```
integer function iran_binomial(N,p)
  implicit none
  integer, intent (in) :: N
  double precision, intent (in) :: p
  double precision :: u,a,F,ran_u
  integer :: m
  a=(1.d0-p)**N
  F=a
  m=0
  u=ran_u()
  do while(F < u)
    a=a*p/(1.d0-p)*(N-m)/(m+1.d0)
    F=F+a
    m=m+1
  enddo
  iran_binomial=m
end function iran_binomial
```

As before, we could start by checking first the value of m equal to the average value of the distribution pN, and then increase or decrease m as needed. In the next chapter, however, we will see alternative methods for the generation of the binomial distribution.

This ends our short introduction to the basic methods for the generation of random variables distributed according to a given probability distribution. We will devote more time to this important topic in forthcoming chapters, but now let us return to the explanation of different integration techniques.

2.5 Importance Sampling

Let us consider again the general integral

$$I = \int g(x)\, dx \tag{2.62}$$

which we write in the form

$$I = \int G(x) f_{\hat{\mathbf{x}}}(x)\, dx \tag{2.63}$$

with $G(x) = \dfrac{g(x)}{f_{\hat{\mathbf{x}}}(x)}$ and $f_{\hat{\mathbf{x}}}(x)$ has to be a probability density function (nonnegative and normalized). In the sampling method, we consider this to be equal to the average $E[G]$ and use the approximation used by the sample mean.

There are infinite ways in which (2.62) can be decomposed as in (2.63), although some might be more ''natural'' than others. For instance, take the integral

$$I = \int_0^\infty dx\, \cos(x) x^2 \mathrm{e}^{-x} \tag{2.64}$$

which is $I = -1/2$. If we would like to use a sampling method for its calculation, possible ''natural'' choices would be

$$G^{(1)}(x) = \cos(x) x^2, \quad x \geq 0, \tag{2.65}$$

$$f_{\hat{\mathbf{x}}}^{(1)}(x) = \begin{cases} 0, & x < 0, \\ e^{-x}, & x \geq 0, \end{cases} \tag{2.66}$$

or

$$G^{(2)} = \cos(x) x, \quad x \geq 0, \tag{2.67}$$

$$f_{\hat{\mathbf{x}}}^{(2)}(x) = \begin{cases} 0, & x < 0, \\ x e^{-x}, & x \geq 0, \end{cases} \tag{2.68}$$

or

$$G^{(3)}(x) = 2\cos(x), \quad x \geq 0, \tag{2.69}$$

$$f_{\hat{\mathbf{x}}}^{(3)}(x) = \begin{cases} 0, & x < 0, \\ \frac{1}{2} x^2 e^{-x}, & x \geq 0, \end{cases} \tag{2.70}$$

but we could have used, for example, a not-so-obvious splitting

$$G^{(4)}(x) = \begin{cases} 0, & x < 0, \\ \pi(x^2+1)\cos(x) x^2 \mathrm{e}^{-x}, & x \geq 0, \end{cases} \tag{2.71}$$

$$f_{\hat{\mathbf{x}}}^{(4)}(x) = \frac{1}{\pi} \frac{1}{1+x^2}, \qquad \forall x. \tag{2.72}$$

Which is the best way to split $g(x) = G(x) f_{\hat{\mathbf{x}}}(x)$? Of course, a possible criterion is that the numerical generation of random numbers according to $f_{\hat{\mathbf{x}}}(x)$ turns out to be easy from a practical point of view. But leaving aside this otherwise very reasonable requirement, a sensible condition to choose the splitting is to obtain an algorithm with the minimum statistical error. If we look at (2.13), the smallest error is obtained when the sample variance $\hat{\boldsymbol{\sigma}}_M^2[G]$ is minimum, or, equivalently, when the variance as given by

$$\sigma^2[G] = \int G[x]^2 f_{\hat{\mathbf{x}}}(x) dx - \left(\int G(x) f_{\hat{\mathbf{x}}}(x) dx \right)^2 = \int \frac{g(x)^2}{f_{\hat{\mathbf{x}}}(x)}\, dx - I^2 \tag{2.73}$$

is a minimum. As I is independent of $f_{\hat{\mathbf{x}}}(x)$, the minimization of $\sigma^2[G]$ is equivalent to the minimization of $\int \frac{g(x)^2}{f_{\hat{\mathbf{x}}}(x)}\,dx$. The minimization has to be achieved in the space of functions $f_{\hat{\mathbf{x}}}(x)$, which can be considered as probability density functions, that is, those functions satisfying

$$f_{\hat{\mathbf{x}}}(x) \geq 0, \tag{2.74}$$

$$\int f_{\hat{\mathbf{x}}}(x)\,dx = 1. \tag{2.75}$$

The optimal solution $f_{\hat{\mathbf{x}}}^{\text{opt}}(x)$ can be found by using the method of the Lagrange multipliers. Let us introduce the functional $\mathcal{L}\left[f_{\hat{\mathbf{x}}}\right]$

$$\mathcal{L}\left[f_{\hat{\mathbf{x}}}\right] = \int \frac{g(x)^2}{f_{\hat{\mathbf{x}}}(x)}\,dx + \lambda \int f_{\hat{\mathbf{x}}}(x)\,dx \tag{2.76}$$

with λ being the Lagrange multiplier needed to take into account the normalization condition (2.75). The minimization of $\mathcal{L}\left[f_{\hat{\mathbf{x}}}\right]$ leads to

$$\left.\frac{\delta \mathcal{L}}{\delta f_{\hat{\mathbf{x}}}}\right|_{f_{\hat{\mathbf{x}}}(x)=f_{\hat{\mathbf{x}}}^{\text{opt}}(x)} = 0 \Longrightarrow -\frac{g(x)^2}{f_{\hat{\mathbf{x}}}^{\text{opt}}(x)^2} + \lambda = 0 \tag{2.77}$$

or

$$f_{\hat{\mathbf{x}}}^{\text{opt}}(x) = +\lambda^{-1/2}\left|g(x)\right|. \tag{2.78}$$

According to condition (2.74), we have taken the + sign for the square root. Now λ is found from the normalization condition (2.75):

$$f_{\hat{\mathbf{x}}}^{\text{opt}}(x) = \frac{\left|g(x)\right|}{\int \left|g(x)\right|\,dx}. \tag{2.79}$$

The corresponding optimal function $G^{\text{opt}}(x)$ is

$$G^{\text{opt}}(x) = \frac{g(x)}{f_{\hat{\mathbf{x}}}^{\text{opt}}(x)} \tag{2.80}$$

and the associated minimum variance is

$$\sigma^2[G^{\text{opt}}] = \left(\int \left|g(x)\right|\,dx\right)^2 - I^2. \tag{2.81}$$

This result indicates that, if $g(x)$ is a nonnegative function, such that $|g(x)| = g(x)$, then the variance of the optimal estimator is 0. In other words, the statistical error is 0 and the sample average is exact (independent of the number of points M used). Where is the trick? If we look at the optimal choice, in the denominator of (2.79) appears precisely the integral I. But knowing the value of the integral was the initial goal. If we know it, why should we need a numerical algorithm whose goal is the calculation of the integral?

However, the idea of importance sampling is to replace the optimal value $f_{\hat{\mathbf{x}}}^{\text{opt}}(x)$ by some other function $f_{\hat{\mathbf{x}}}(x)$ close to it (while still keeping the generation of the

random variable easy enough). For example, let us look at the calculation of the integral in (2.64). The optimal choice would be the splitting

$$G^{\rm opt}(x) = \lambda^{1/2}\frac{\cos(x)}{|\cos(x)|}, \quad x \geq 0, \tag{2.82}$$

$$f_{\tilde{\mathbf{x}}}^{\rm opt}(x) = \begin{cases} 0, & x < 0, \\ \lambda^{-1/2}|\cos(x)|x^2 e^{-x}, & x \geq 0 \end{cases} \tag{2.83}$$

with $\lambda^{1/2} = \int_0^\infty |\cos(x)|x^2 e^{-x}dx$ being the normalization constant.[6] The minimal variance of the optimal choice is $\sigma^2[G^{\rm opt}] = \lambda - I^2 \approx 1.190009$. However, there is no simple way to solve $\int_0^x f_{\tilde{\mathbf{x}}}^{\rm opt}(x')dx' = u$ and the optimal choice is useless. We could use instead, for example, any of the four splittings given in (2.65)–(2.71). Which one would be the most efficient? The one that uses a probability density function $f_{\tilde{\mathbf{x}}}(x)$ closer to $f_{\tilde{\mathbf{x}}}^{\rm opt}$. Intuitively, it is option number 3, as all it does is to replace $|\cos(x)|$ by an average value $1/2$. We can check this out by computing (analytically) the variance in the four cases using the integrals of (2.73), with the result

$$\sigma^2[G^{(1)}] = \frac{148843}{12500} \approx 11.907,$$
$$\sigma^2[G^{(2)}] = \frac{6791}{2500} \approx 2.716,$$
$$\sigma^2[G^{(3)}] = \frac{787}{500} \approx 1.574,$$
$$\sigma^2[G^{(4)}] = -\frac{1}{4} + \frac{27\pi}{16} \approx 5.051.$$

Hence, the splitting number 3, given by (2.69), is indeed the most efficient, at least from the point of view of the associated variance. Of course, we need to check that the generation of the random numbers distributed according to $f_{\tilde{\mathbf{x}}}^{(3)}(x)$, a $\hat{\mathbf{\Gamma}}(3,1)$ distribution, is not so slow as to render the method more inefficient than the others. We will return to this point later in the chapter.

Once we have determined which is the optimal splitting, all we need to do is to use subroutine `mc3` with a function $G^{(3)}(x) = 2\cos(x)$ and `ran_gamma3` for the generation of random numbers distributed according to the $\hat{\mathbf{\Gamma}}(3,1)$ distribution. For the sake of clarity, we include below a driver program as well as an implementation of the function $G^{(3)}(x)$, which together with subroutine `mc3` and function `ran_gamma3` leads to a full program.

```
program area2
  implicit none
  interface
    double precision function g3(x)
      double precision, intent (in) :: x
    end function g3
  end interface
  interface
```

6) The integration can be performed analytically to obtain
$\lambda^{1/2} = \frac{2-6e^{\pi}+6e^{2\pi}-2e^{3\pi}+e^{\pi/2}(-2+\pi)^2+e^{5\pi/2}(2+\pi)^2+e^{3\pi/2}(-8+6\pi^2)}{4(-1+e^{\pi})^3} \approx 1.20003565\ldots.$

```
    double precision function ran_gamma3()
    end function ran_gamma3
  end interface
  double precision :: r,s
  integer :: M
  M=100000
  call mc3(g3,ran_gamma3,M,r,s)
  write (*,*) "The estimated value of the integral is", r
  write (*,*) "The estimated error is", s
end program area2

double precision function g3(x)
  implicit none
  double precision, intent (in) ::x
  g3=2.d0*cos(x)
end function g3
```

The reader should run this program with a reasonably large number, for example, $M = 10^5$, and check that the numerical result agrees, within errors, with the known value $I = -1/2$.

Summing up, the basic idea of the importance sampling method is to split the integrand as $g(x) = G(x)f_{\hat{\mathbf{x}}}(x)$ using a suitable random variable $\hat{\mathbf{x}}$ with pdf $f_{\hat{\mathbf{x}}}(x)$. It is desirable to chose $f_{\hat{\mathbf{x}}}(x)$ in such a way that (i) it is reasonably simple to generate, and (ii) it gives a small variance. The optimal $f_{\hat{\mathbf{x}}}(x)$ most often does not satisfy condition (i), but we might then look for functions $f_{\hat{\mathbf{x}}}(x)$ that are close enough to the optimal one. For example, let us consider the integral of $g(x) = J_0\left(x_0 x\right)$

$$I = \int_0^1 dx J_0\left(x_0 x\right) \tag{2.84}$$

where J_0 is the Bessel function and $x_0 = 2.404825557695773\ldots$ the first zero of $J_0(x)$. Looking at the shape of J_0, we propose the family of pdfs

$$f_{\hat{\mathbf{x}}}(x) = \frac{6}{3+a}\left(-ax^2 + (a-1)x + 1\right) \qquad x \in (0,1) \tag{2.85}$$

which is a properly normalized family of parabolic functions that have a maximum at $x = 0$ if $a < 1$ and vanish at $x = 1$. Positivity is ensured if $a \geq -1$. Therefore, a is restricted to the interval $(-1, 1)$. We define, hence

$$G(x) = \frac{g(x)}{f_{\hat{\mathbf{x}}}(x)} = \frac{3+a}{6}\frac{J_0\left(x_0 x\right)}{\left(-ax^2 + (a-1)x + 1\right)} \tag{2.86}$$

and use the sample mean of $G(x)$ as an unbiased estimator for the integral. To generate random numbers distributed according to $f_{\hat{\mathbf{x}}}(x)$, we need to invert the cumulative distribution function

$$F_{\hat{\mathbf{x}}}(x) = \frac{6}{3+a}\left(-\frac{a}{3}x^3 + \frac{a-1}{2}x^2 + x\right). \qquad x \in (0,1) \tag{2.87}$$

Although in this case Cardano's formula gives us the solution of the cubic equation $F_{\hat{\mathbf{x}}}(x) = u$, it is actually easier to implement again the Newton–Raphson algorithm. The following program returns random numbers distributed according to $f_{\hat{\mathbf{x}}}(x)$.

```
double precision function ran_poli2()
  implicit none
  double precision :: a,x,xn,fx,fxc,u,ran_u
  common /a/a
    fx(x)=6.0d0/(3.0d0+a)*(1.0d0+x*(a-1.0d0-a*x))
    fxc(x)=6.0d0/(3.0d0+a)*x*(1.0d0+x*((a-1.0d0)/2.0d0-a/3.0d0*x))
    u=ran_u()
    x=u
    xn=x-(fxc(x)-u)/fx(x)
    do while(abs(xn-x) > 1.0d-8)
      xn=x
      x=x-(fxc(x)-u)/fx(x)
    enddo
    ran_poli2=x
end function ran_poli2
```

We give here an example of a driver and an implementation of $G(x)$:

```
program area3
  implicit none
  interface
    double precision function gb(x)
      double precision, intent (in) :: x
    end function gb
  end interface
  interface
    double precision function ran_poli2()
    end function ran_poli2
  end interface
  double precision :: a,r,s
  integer :: M
  common /a/a
  M=1000000
  a=0.5d0
  call mc3(gb,ran_poli2,M,r,s)
  write (*,*) "The estimated value of the integral is", r
  write (*,*) "The estimated error is", s
end program area3

double precision function gb(x)
  implicit none
  double precision, intent (in) :: x
  double precision, parameter :: x0=2.404825557695773d0
  double precision :: a,fx,dbesj0
  common /a/a
  fx(x)=6.d0/(3.d0+a)*(1.d0+x*(a-1.d0-a*x))
  gb=dbesj0(x*x0)/fx(x)
end function gb
```

This has to be complemented with the function `ran_poli2()` and the subroutine `mc3`. Here, `dbesj0` is the Bessel function $J_0(x)$ as implemented in gfortran

and also in the Intel fortran compiler. Otherwise, a routine that returns the value of the Bessel function $J_0(x)$ must be provided. On running this program for different values of a, it is found that the minimal error is obtained for $a \approx 0.4$. From a particular run, we have obtained[7] $I = 0.611411 \pm 0.000024$. We do not have an exact value to compare it with now, but a numerical integration using more traditional techniques[8] yields $I = 0.6113957$. The difference is 1.5×10^{-5}, in perfect agreement with the importance sampling estimate, including the error.

2.6 Advantages of Monte Carlo Integration

The reader might be somehow disappointed as the examples we have given so far could be done either analytically, for example, (2.64), or by other numerical methods with more precision, in the case of (2.84). Although there might be examples of one-dimensional integrals in which the Monte Carlo integration could be competitive compared to more traditional methods (like Simpson integration), the truth is that the real power of Monte Carlo integration lies in the calculation of N-dimensional integrals. Let us consider, for example, the integral

$$I(N) = \int_{-\infty}^{\infty} dx_1 \cdots \int_{-\infty}^{\infty} dx_N e^{-(x_1+\cdots+x_N)} J_0\left(x_1^2 + \cdots + x_N^2\right). \tag{2.88}$$

For large N, it would be difficult to write an efficient code implementing, for instance, a Simpson-like algorithm. Let us consider it from the point of view of importance sampling. We split the integrand $g(x_1, \ldots, x_N) = G(x_1, \ldots, x_N) f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_N)$ with

$$G(x_1, \ldots, x_N) = J_0\left(x_1^2 + \cdots + x_N^2\right), \tag{2.89}$$

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_N) = e^{-(x_1+\cdots+x_N)} = e^{-x_1} \cdots e^{-x_N}. \tag{2.90}$$

The key point is that $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_N)$ can be considered as the probability density function of an N-dimensional random variable $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N)$. In fact, as it can be factorized as $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_N) = f_{\hat{\mathbf{x}}}(x_1) \cdots f_{\hat{\mathbf{x}}}(x_N)$ with $f_{\hat{\mathbf{x}}}(x) = e^{-x}$, the N random variables $\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N$ are independent of each other and follow the same exponential distribution. It is very easy to use the method of importance sampling to compute (2.88): generate M values $\mathbf{x}_1, \ldots, \mathbf{x}_M$ of the N-dimensional vector $\mathbf{x} = (x_1, \ldots, x_N)$; use them to compute the sample mean and variance of $G(\mathbf{x})$, using the known formulas (2.11) and (2.12); and approximate the integral by (2.13).

7) Needless to say, running the program with different random number generators will produce different results. This value is just an example of one output of this program, not the one the reader will obtain when running the algorithm by him/herself. This comment applies to all the cases in which we give a numerical estimate of an integration using any of these Monte Carlo methods.

8) Actually, it is the result given by the Mathematica program.

Let us present a simple computer program:

```
program n_dimensional_integral
  implicit none
  interface
    double precision function ran_exp(a)
      double precision, intent (in) :: a
    end function ran_exp
  end interface
  double precision :: r,s,x,dbesj0,sx,s2,g0
  integer, parameter :: N=4
  integer ::M,i,k
  M=1000000
    r=0.d0
    s=0.d0
    do k=1,M
      s2=0.d0
      do i=1,N
        x=ran_exp(1.d0)
        s2=s2+x*x
      enddo
      g0=dbesj0(s2)
      r=r+g0
      s=s+g0*g0
    enddo
    r=r/M
    s=sqrt((s/M-r*r)/M)
  write (*,*) "The estimated value of the integral is", r
  write (*,*) "The estimated error is", s
end program n_dimensional_integral
```

As an example, we have run this program with $M = 10^6$ samples and it took us a fraction of a second on a desktop computer to obtain the values $I(2) = 0.38596 \pm 0.00049$, $I(3) = 0.20028 \pm 0.00044$, and $I(4) = 0.08920 \pm 0.00036$. These are in agreement with what you can get with other numerical methods, but are much faster to obtain. The complexity of the Monte Carlo algorithm does not increase with N, the number of integration variables. The program runs perfectly for $N = 10$ giving, using $M = 10^8$, $I(10) = -0.002728 \pm 0.000016$ in less than 1 min, whereas it would be extremely costly to get the same accuracy with a deterministic integration algorithm.

2.7 Monte Carlo Importance Sampling for Sums

The same ideas of the different Monte Carlo integration techniques can be used in the case of sums. Imagine we want to compute a sum $\sum_i g_i$ and that it is possible

to split $g_i = G_i p_i$ in such a way that $p_i \geq 0$ and $\sum_i p_i = 1$. Then, the sum can be considered as the average value of a random variable $\hat{\mathbf{x}}$ that takes only integer values, such that its pdf is

$$f_{\hat{\mathbf{x}}}(x) = \sum_i p_i \delta(x - i). \tag{2.91}$$

We can then consider this sum as the average value $E[G]$ and evaluate it using the sample mean. In this case, this means the generation of integer values $i_1, \ldots, i_M$ distributed with the set of probabilities p_i, and then computing the sample mean and variance:

$$\hat{\boldsymbol{\mu}}_M[G] = \frac{1}{M} \sum_{k=1}^{M} G_{i_k} \tag{2.92}$$

$$\hat{\boldsymbol{\sigma}}^2_M[G] = \frac{1}{M} \sum_{k=1}^{M} G^2_{i_k} - \left(\frac{1}{M} \sum_{k=1}^{M} G_{i_k} \right)^2 . \tag{2.93}$$

For example, if we want to compute the sum

$$S = \sum_{i=0}^{\infty} i^{1/2} 2^{-i} \tag{2.94}$$

(the exact value is $\text{PolyLog}\left[-\frac{1}{2}, \frac{1}{2}\right] = 1.347253753\ldots$), we split $g_i = G_i p_i$ with $G_i = 2i^{1/2}$ and $p_i = \frac{1}{2} 2^{-i}$. Then the p_i's are the probabilities of a discrete geometric distribution of parameter $q = 1/2$ and we already learned in Section 2.4 how to generate those. A full program is as follows:

```
program monte_carlo_sum
  implicit none
  interface
    integer function iran_geo(a)
       double precision, intent (in) :: a
    end function ran_exp
  end interface
  double precision :: q,r,s,a,g0
  integer :: M,i,k
  p=0.5d0
  M=1000000
  r=0.d0
  s=0.d0
  a=-log(q)
  do k=1,M
    i=iran_geo(p)
    g0=2.d0*sqrt(dble(i))
    r=r+g0
    s=s+g0*g0
  enddo
  r=r/M
  s=sqrt((s/M-r*r)/M)
```

```
  write (*,*) "The estimated value of the sum is", r
  write (*,*) "The estimated error is", s
end program monte_carlo_sum
```

A particular run of this program with $M = 10^8$ gave us $S = 1.34714 \pm 0.00015$, in perfect agreement, within statistical errors, with the exact result. Again, the real advantage of the Monte Carlo integration lies in the numerical calculations of high-dimensional sums. For example, the sum

$$S(N) = \sum_{i_1=0}^{\infty} \cdots \sum_{i_N=0}^{\infty} \frac{2^{-(i_1+\cdots+i_N)}}{1+i_1^2+\cdots+i_N^2} \tag{2.95}$$

can be evaluated by splitting $g_{i_1\cdots i_N} = G_{i_1\cdots i_N} p_{i_1\cdots i_N}$, with $G_{i_1\cdots i_N} = \frac{2^N}{1+i_1^2+\cdots+i_N^2}$ and $p_{i_1\cdots i_N} = \left(\frac{1}{2}2^{-i_1}\right)\cdots\left(\frac{1}{2}2^{-i_N}\right)$, the product of N independent geometric distributions. The program is as follows:

```
program monte_carlo_multidimensional_sum
  implicit none
  interface
    integer function iran_geo(a)
       double precision, intent (in) :: a
    end function ran_exp
  end interface
  integer, parameter :: N=4
  double precision :: q,r,s,sx,a,g0
  integer :: M,i,k,j
  p=0.5d0
  M=100000000
  r=0.d0
  s=0.d0
  a=-log(q)
  do k=1,M
    sx=1.0d0
    do i=1,N
      j=iran_geo(p)
      sx=sx+j*j
    enddo
    g0=2.0d0**N/sx
    r=r+g0
    s=s+g0*g0
  enddo
  r=r/M
  s=sqrt((s/M-r*r)/M)
  write (*,*) "The estimated value of the sum is", r
  write (*,*) "The estimated error is", s
end program monte_carlo_multidimensional_sum
```

A particular run with $M = 10^8$ gave the following results: $S(1) = 1.318107 \pm 0.000073$, $S(2) = 1.79352 \pm 0.00014$, $S(4) = 3.67076 \pm 0.00040$,

and $S(10) = 63.5854 \pm 0.0071$. While the first three are in agreement with other numerical methods to evaluate sums, the last result, for $N = 10$, which requires less than 1 min on a desktop computer, would be difficult to evaluate using any other method.

2.8 Efficiency of an Integration Method

This is a concept very easy to understand. The efficiency of an integration method is measured by the time it takes for a computer to provide an estimate of the integral with some given error ϵ. In the Monte Carlo methods explained so far, the error is always given by

$$\epsilon = \frac{\sigma}{\sqrt{M}} \tag{2.96}$$

with σ, the standard deviation, being an intrinsic value independent of the number of repetitions M. If we fix the error ϵ, it turns out that $M = \sigma^2/\epsilon^2$ is proportional to the variance of the estimator, σ^2. At first sight, then, it seems natural to use an estimator with the smallest possible variance (i.e., the importance sampling), but there is another factor to take into consideration: the actual time t (in seconds) that it takes to generate each contribution to the estimator. This consists of the time spent in every call to the function `ran_f()` plus the time needed to compute the function $G(x)$ and do the sums contributing to the average and the variance. The total needed time is then Mt, which is proportional to $t\sigma^2$. Hence, it might pay off to use a method with a larger variance if the time needed to generate each contribution to the sample mean compensates the larger variance. The relative efficiency between integration methods 1 and 2 is nothing but the ratio

$$e_{12} = \frac{t_1\sigma_1^2}{t_2\sigma_2^2} \tag{2.97}$$

with t_1 and σ_1^2 being the time and variance of method 1, and similarly for method 2. If $e_{12} > 1$, we conclude that method 2 is more efficient than method 1.

We can prove, for instance, that the method of uniform sampling is more efficient than the hit-and-miss method. For the latter method, recalling (2.4) and (2.5), we get for the variance of this method as

$$\sigma_1^2 = c(b-a)I - I^2. \tag{2.98}$$

For the uniform sampling method, on the other hand, the variance is

$$\sigma_2^2 = (b-a)\int g(x)^2\,dx - I^2. \tag{2.99}$$

So

$$\sigma_1^2 - \sigma_2^2 = (b-a)\left[cI - \int g(x)^2\,dx\right]. \tag{2.100}$$

Since we had the condition $0 \leq g(x) \leq c$, we obtain

$$\int g(x)^2\, dx \leq c \int g(x)\, dx = cI \tag{2.101}$$

and $\sigma_1 \geq \sigma_2$. Furthermore, given that the hit-and-miss method requires the calculation of two random numbers whereas the uniform sampling requires only one, it turns out that the former takes more time, $t_1 > t_2$. As $e_{12} > 1$, we conclude that uniform sampling is more efficient than the hit-and-miss method.

2.9 Final Remarks

In this chapter, we have argued that numerical integration based on the sampling algorithm

$$\int G(\mathbf{x}) f_{\hat{\mathbf{x}}}(\mathbf{x}) d\mathbf{x} = \hat{\boldsymbol{\mu}}_M[G] \pm \frac{\hat{\boldsymbol{\sigma}}_M[G]}{\sqrt{M}} \tag{2.102}$$

with

$$\hat{\boldsymbol{\mu}}_M[G] = \frac{1}{M} \sum_{k=1}^{M} G(\mathbf{x}_k), \tag{2.103}$$

$$\hat{\boldsymbol{\sigma}}_M^2[G] = \frac{1}{M} \sum_{k=1}^{M} G(\mathbf{x}_k)^2 - \left(\frac{1}{M} \sum_{k=1}^{M} G(\mathbf{x}_k) \right)^2 \tag{2.104}$$

where $\mathbf{x}_k, k = 1, \ldots, M$ are the values of the random variable $\hat{\mathbf{x}}$ whose pdf is $f_{\hat{\mathbf{x}}}(\mathbf{x})$, can be very competitive if $\mathbf{x} = (x_1, \ldots, x_N)$ is high dimensional. How high is high? How large must N be for this method to be competitive? The exact answer might depend on the specific details of the function $G(\mathbf{x})$ and the difficulty in the generation of the random variables. The answer might be $N > 10$ or $N > 5$, but one thing is sure: when N is very large, on the order of thousands or millions of variables involved in the integral, then *there is no alternative to the Monte Carlo sampling*. There are many problems that need of the calculation of such high-dimensional integrals, but the typical applications are in the field of statistical mechanics, where ideally one would like to deal with N on the order of the Avogadro number, $N \sim 10^{23}$, although we are very far away from being able to get close to this number with today's (or tomorrow's, for that matter) computer capabilities. We have reached a stage, however, where it is not unusual to consider N to be in the range of 10^6, although much larger values can be dealt with satisfactorily in some specific cases.

It is clear that, to proceed, we need efficient ways of generating the values of the N-dimensional random variable $\hat{\mathbf{x}} = (\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N)$ whose pdf is $f_{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N}(x_1, \ldots, x_N)$. In the few examples of this chapter in which we have used the sampling method for N-dimensional integrals or sums, we have managed to split $f_{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_N}(x_1, \ldots, x_N) = f_{\hat{\mathbf{x}}}(x_1) \ldots f_{\hat{\mathbf{x}}}(x_N)$ as the product of N identical and independent random variables. Unfortunately, this it not the case of most applications of interest and we must

develop methods to generate those high-dimensional random variables. Before we dwell into this, we will explain further algorithms valid for one-variable pdfs and then move on to high-dimensional variables.

Further Reading and References

The basic Monte Carlo integration techniques, including hit and miss, sampling methods, and others not explained here, can be found in the classic books by Kalos and Whitlock [3] and Rubinstein [4].

Exercises

1) Compute numerically, using the hit-and-miss Monte Carlo method, the integral

$$\int_0^1 dx\sqrt{1-x^2}.$$

Compute the dependence of the standard deviation σ of the Monte Carlo estimator with the number of points used (take $M = 10, 100, 1000$, etc.). Calculate the integral analytically and plot in a log–log scale the real error (absolute difference between the exact value and the Monte Carlo estimator) versus N. What is the slope of the curve as $M \to \infty$?

2) Repeat the previous problem using the uniform sampling method. What is the relation between the standard deviations σ of both methods? Measure the time it takes to run both algorithms and find which method is more efficient. How many CPU seconds will it take to compute the integral with an error less than 10^{-6}?

3) Repeat the two previous problems 10^4 times with $M = 100$ using each time a different sequence of random numbers. Compute, for each method, the percentage of cases for which the numerical result differs from the exact one in less than (a) 1σ, (b) 2σ, and (c) 3σ.

4) Compute the integral of the previous exercises using the method of uniform sampling with $M = 10^4$. Compute the standard deviation σ of the result. Let μ_1 and σ_1 be, respectively, the estimator and the standard deviation obtained. Repeat 10 times using each a different sequence of random numbers and obtain 10 different estimators $\mu_1, \ldots, \mu_{10}$ and standard deviation $\sigma_1, \ldots, \sigma_{10}$. Compute a new estimator μ and standard deviation σ from the average of the 10 values I_i and their standard deviation. What relation do you expect between I_i, σ_i, I, and σ? Check that relation.

5) Using the pdf $f_{\hat{\mathbf{x}}}(x) = \exp(-x)$, $x \geq 0$, compute using the sampling method the integral

$$\int_0^\infty dx\sqrt{x}\cos(x)\exp(-x).$$

6) Compute the integral

$$I = \int_0^1 dx \cos\left(\frac{\pi x}{2}\right)$$

using uniform sampling and a general sampling method with the pdf $f_{\hat{\mathbf{x}}}(x) = \frac{3}{2}(1 - x^2)$. What is the relation between the errors of both methods? What is their relative efficiency?

7) Compute the integral of the previous problems using the sampling method with the pdf $f_{\hat{\mathbf{x}}}(x) = \frac{6}{3+a}[1 + x(a-1) - ax^2]$. Compute numerically the standard deviation of the result as a function of a and determine the optimal value of a. Compare the efficiency for $a = 1$ (as used in the previous problem) and the optimal value.

8) Compute using the sampling method the integral of problem 1 using the pdf $f_{\hat{\mathbf{x}}}(x) = \frac{1-ax^2}{1-a/3}$ depending on the parameter a and compute, numerically, the optimal value for a.

9) For problem 1, check that, when using the optimal pdf $f_{\hat{\mathbf{x}}}^{\text{opt}}(x)$ as given by the importance sampling method, the sampling error is zero whatever the number of points used for the numerical integration.

10) Compute

$$\int_0^1 dx_1 \ldots \int_0^1 dx_n \exp\left[-\frac{1}{2}\left(x_1^2 + x_2^2 + \cdots + x_n^2\right)\right] \cos^2\left(x_1x_2 + x_2x_3 + \cdots + x_nx_1\right)$$

for $n = 1, 2, 3, 5, 10$ using Simpson's, uniform sampling, and hit-and-miss methods.

11) Prove, without using variational methods, that the function $f_{\hat{\mathbf{x}}}^{\text{opt}}$ is the one that minimizes the rms of the numerical estimator to the integral I. Show that this is equivalent to proving

$$\left(\int |g(x)|\, dx\right)^2 \leq \int \frac{g(x)^2}{f_{\hat{\mathbf{x}}}(x)}\, dx$$

and this follows from Schwartz's inequality

$$\left(\int f_1(x) f_2(x)\, dx\right)^2 \leq \int f_1(x)^2\, dx \int f_2(x)^2\, dx.$$

3
Generation of Nonuniform Random Numbers: Noncorrelated Values

3.1
General Method

We have already described in Section 2.4 a method that allows, in principle, the generation of values of a random variable distributed according to a given probability distribution function $f_{\hat{\mathbf{x}}}(x)$ based on the relation $\hat{\mathbf{x}} = F^{-1}_{\hat{\mathbf{x}}}(\hat{\mathbf{u}})$, with $\hat{\mathbf{u}}$ being a $\hat{\mathbf{U}}(0,1)$ random variable uniformly distributed in the interval $(0,1)$.

The main problem with this general method is the technical difficulty in finding a good implementation of the inverse cumulative distribution function $F^{-1}_{\hat{\mathbf{x}}}$. When this happens, a possibility is to use the Newton–Raphson method to find the solution of $F_{\hat{\mathbf{x}}}(x) - u = 0$. As $F'_{\hat{\mathbf{x}}}(x) = f_{\hat{\mathbf{x}}}(x)$, the algorithm proceeds by iteration as described in (2.39). After setting an initial value x_0, the recurrence proceeds until $|x_{n+1} - x_n| < \epsilon$, a prefixed accuracy, for instance, $\epsilon = 10^{-8}$. However, this method might be slow, and it is not guaranteed that the algorithm always gives the solution of the equation because it is known that the Newton–Raphson algorithm might not converge. Therefore, one needs to do a good deal of testing before the algorithm can actually be used with any guarantee of success. Of course, this method needs a good routine for the calculation of $F_{\hat{\mathbf{x}}}(x)$, which is not always the case. Imagine we want to apply it to generate random numbers distributed according to the gamma family of distributions.[1] The pdf of the $\hat{\mathbf{\Gamma}}(\alpha)$ distribution is given in (1.59). The cumulative function is

$$F_{\hat{\mathbf{x}}}(x;\alpha) = \int_{-\infty}^{x} dx' f_{\hat{\mathbf{x}}}(x') = \begin{cases} 0, & x < 0, \\ \dfrac{\gamma(\alpha;x)}{\Gamma(\alpha)}, & x \geq 0, \end{cases} \tag{3.1}$$

$\gamma(\alpha;x)$ being the lower incomplete gamma function.[2] To generate random numbers x distributed according to the gamma distribution using the inverse cdf $x = F^{-1}_{\hat{\mathbf{x}}}(u)$, all we need, then, is a good algorithm to compute the inverse lower incomplete gamma function $\gamma^{-1}(\alpha;x)$. Alas, this function is not one of the standard functions

1) We are simplifying the notation and denote $\hat{\mathbf{\Gamma}}(\alpha) \equiv \hat{\mathbf{\Gamma}}(\alpha, \theta = 1)$. To generate a random variable distributed according to $\hat{\mathbf{\Gamma}}(\alpha,\theta)$, we simply multiply by θ a random variable distributed according to $\hat{\mathbf{\Gamma}}(\alpha)$.

2) This is defined precisely as $\gamma(\alpha;x) = \int_0^x ds\, s^{\alpha-1}\mathrm{e}^{-s}$.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

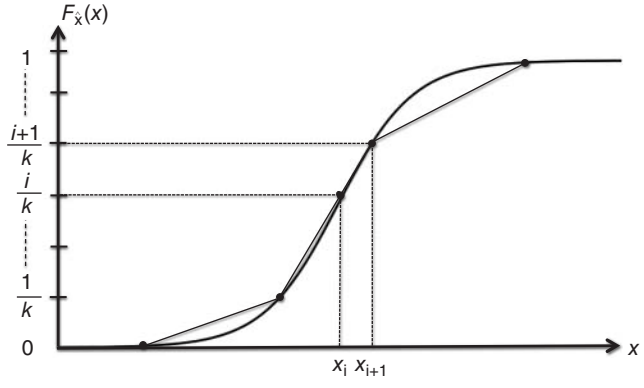


Figure 3.1 $F_{\hat{\mathbf{x}}}(x)$ and the piecewise linear approximation used in the numerical inversion method.

you get "for free" in the most popular compilers, and you might need to shop around to find a good library that implements this function. The same technical difficulty arises when finding the numerical solution of $F_{\hat{\mathbf{x}}}(x) = u$ using, for instance, the Newton–Raphson method, because the lower incomplete gamma function itself, $\gamma(\alpha; x)$, is not standard. Even if we manage to find suitable implementations for $\gamma(\alpha; x)$ or $\gamma^{-1}(\alpha; x)$ (or if we write those ourselves), it is more likely that the resulting algorithm will be slow as it will require very many calculations.

A good alternative to solving $F_{\hat{\mathbf{x}}}(x) = u$ every time we need a random number x might be to use a numerical inversion algorithm. The trick is to divide the $[0, 1]$ interval into K subintervals $\left[\frac{i}{K}, \frac{i+1}{K}\right]_{i=0,1,\ldots,K-1}$ and tabulate the inverse cumulative distribution function at the points $u_i = i/K$, that is, compute $x_i = F^{-1}(i/K)$ for $i = 0, \ldots, K$. However painful this might be, the good news is that you need to do it only once. The set of numbers x_i can be kept in a file and read whenever they are needed. The next step is to replace the true $F_{\hat{\mathbf{x}}}(x)$ by its piecewise-linear approximation in the interval $[\frac{i}{K}, \frac{i+1}{K}]$, see Figure 3.1. Given a number $u \in [0, 1]$, it is straightforward to invert the linear interpolation to $x = F^{-1}_{\hat{\mathbf{x}}}(u)$ by means of

$$x = (Ku - i)x_{i+1} + (i + 1 - Ku)x_i \tag{3.2}$$

where i is such that the number u belongs to the interval $[\frac{i}{K}, \frac{i+1}{K})$, or $i = [Ku]$ (the integer part of Ku). A possible program could be as follows:

```
double precision function ran_f(x,K)
implicit none
double precision (:) :: x
double precision u,ran_u
integer K,i
```

```
u=ran_u()
i=K*u
ran_f=(K*u-i)*x(i+1)+(i+1-K*u)*x(i)
```

```
end function ran_f
```

Recall that, in the Fortran language, there is an automatic truncation for integer numbers. So, the line `i=K*u` is equivalent to `i=int(K*u)`. Before the first call of this routine, we need to initialize it by choosing a value for K and filling the entries of `x(i)` for $i = 0, \ldots, K$ with the values $x_i = F_{\hat{\mathbf{x}}}^{-1}(i/K)$. A technical problem arises if either $x_0 = -\infty$ or $x_K = +\infty$, as it happens in common distributions. The usual procedure is then to set $x_0 = -\Gamma_0$ and $x_K = +\Gamma_1$, with Γ_0 and Γ_1 being the cut-off values. One then has to check whether these cut-off values correspond to events of very low probability and can be safely discarded. Another possibility is not to cut off the distribution, but to use the numerical inversion only for the part of the distribution $f_{\hat{\mathbf{x}}}(x)$ limited to $x \in (-\Gamma_0, \Gamma_1)$ and to use another exact or approximate method for values of x outside this interval.

Whether this numerical solution to the inverse cumulative function will yield good-quality random numbers depends on the goodness of the piecewise approximation to $F_{\hat{\mathbf{x}}}(x)$. This is, in turn, determined by the smoothness of $F_{\hat{\mathbf{x}}}(x)$ and by the number of subdivisions K of the $[0, 1]$ interval. On the other hand, once the table of x_i's has been generated, the method is very quick, as it only involves sums and multiplications.

3.2 Change of Variables

Sometimes, a random variable that is difficult to generate can become easy after a change of variables. We gave in (1.23) the relation between the pdfs of two random variables $\hat{\mathbf{y}}$ and $\hat{\mathbf{x}}$ related by a known function $\hat{\mathbf{y}} = y(\hat{\mathbf{x}})$. A very simple case is the linear change $\hat{\mathbf{y}} = a\hat{\mathbf{x}} + b$ with $a \neq 0$. The only solution of $y = ax + b$ is $x = (y - b)/a$ and the pdf of $\hat{\mathbf{y}}$ is

$$f_{\hat{\mathbf{y}}}(y) = \frac{1}{|a|} f_{\hat{\mathbf{x}}}\left(\frac{y-b}{a}\right). \tag{3.3}$$

For example, we have already used that, if $\hat{\mathbf{x}}$ is a $\hat{\mathbf{U}}(0, 1)$, then $\hat{\mathbf{y}}$ is $\hat{\mathbf{U}}(b, a + b)$ for $a > 0$ or $\hat{\mathbf{U}}(a + b, b)$ for $a < 0$, as well as the fact that, if $\hat{\mathbf{x}}$ is a Gaussian $\hat{\mathbf{G}}(0, 1)$ variable, then the change $\hat{\mathbf{y}} = \sigma\hat{\mathbf{x}} + \mu$ converts $\hat{\mathbf{y}}$ into a $\hat{\mathbf{G}}(\mu, \sigma)$ Gaussian variable.

It is possible to derive useful algorithms using changes of variables in more than one dimension. An interesting example appears when we consider two independent Gaussian $\hat{\mathbf{G}}(0, 1)$ random variables $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$, such that the joint probability density function is

$$f_{\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2}(x_1, x_2) = f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x_2) = \frac{1}{2\pi} e^{-\frac{x_1^2 + x_2^2}{2}}. \tag{3.4}$$

Consider the change to polar coordinates $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$:

$$\begin{aligned} \hat{\mathbf{x}}_1 &= \hat{\mathbf{r}} \cos(\hat{\boldsymbol{\theta}}), \\ \hat{\mathbf{x}}_2 &= \hat{\mathbf{r}} \sin(\hat{\boldsymbol{\theta}}). \end{aligned} \tag{3.5}$$

We now apply (1.79) to this change of variables. As for a given x_1, x_2 there is a unique pair of values (r, θ), the joint pdf of $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$ is

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\theta}}}(r,\theta) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)}{J\left(\frac{r,\theta}{x_1,x_2}\right)}. \tag{3.6}$$

The Jacobian is $1/r$ and, after replacing $r^2 = x_1^2 + x_2^2$, we obtain

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\theta}}}(r,\theta) = \frac{1}{2\pi} r e^{-\frac{r^2}{2}}. \tag{3.7}$$

This can be written in the form

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\theta}}}(r,\theta) = f_{\hat{\mathbf{r}}}(r) f_{\hat{\boldsymbol{\theta}}}(\theta) \tag{3.8}$$

with

$$f_{\hat{\mathbf{r}}}(r) = r e^{-\frac{r^2}{2}}, \tag{3.9}$$

$$f_{\hat{\boldsymbol{\theta}}}(\theta) = \frac{1}{2\pi} \tag{3.10}$$

which shows that $\hat{\mathbf{r}}$ and $\hat{\boldsymbol{\theta}}$ are also independent random variables. More specifically, $\hat{\boldsymbol{\theta}}$ is uniformly distributed in the interval $[0, 2\pi]$, which can be obtained simply as $\hat{\boldsymbol{\theta}} = 2\pi\hat{\mathbf{u}}$, with $\hat{\mathbf{u}}$ being a uniform $\hat{\mathbf{U}}(0,1)$ variable. $\hat{\mathbf{r}}$ follows a Rayleigh distribution, and we have already shown in Section 2.4 that it can be generated by $\hat{\mathbf{r}} = \sqrt{-2\log(\hat{\mathbf{v}})}$. This means that the variables $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ can be generated by

$$\begin{aligned} \hat{\mathbf{x}}_1 &= \sqrt{-2\log(\hat{\mathbf{v}})}\cos(2\pi\hat{\mathbf{u}}), \\ \hat{\mathbf{x}}_2 &= \sqrt{-2\log(\hat{\mathbf{v}})}\sin(2\pi\hat{\mathbf{u}}). \end{aligned} \tag{3.11}$$

This way of obtaining two independent Gaussian $\hat{\mathbf{G}}(0,1)$ variables $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ starting from two independent uniform $\hat{\mathbf{U}}(0,1)$ variables $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ is the celebrated Box–Muller–Wiener algorithm. The main advantage of this algorithm, besides being exact and not relying on approximations to the inverse error function, is that it uses only common functions (sine, cosine, log, square-root) that can be found in (almost) any computing language. At the same time, it turns out to be somewhat slow because these functions are computed by complicated routines (it easily takes hundreds of times longer to compute a sine function than to multiply two numbers). It all depends on what you need. If you just need a few thousands of Gaussian random numbers, then use the Box–Muller–Wiener algorithm; if your needs are in the millions of Gaussian random numbers, you might want to implement a more efficient routine based on numerical inversion, for example. We give now a possible implementation of the Box–Muller–Wiener algorithm. We need to know whether the function being called for is an even or odd number of times, since the second time of a pair we need not generate again the uniform random numbers u, v.

```
double precision function ran_gbmw()
 implicit none
 double precision ran_u
 double precision, parameter :: pi2=6.283185307179586d0
 double precision, save :: u,v
 logical, save :: icount=.true.
```

```
  if (icount) then
    u=dsqrt(-2.0d0*log(ran_u()))
    v=pi2*ran_u()
    ran_gbmw=u*cos(v)
    icount=.false.
  else
    ran_gbmw=u*sin(v)
    icount=.true.
  endif
```

```
end function ran_gbmw
```

The change of variables does not need to lead from a set of n random variables to another set of exactly the same number of variables. For instance, we could consider the change $\hat{\mathbf{x}} = \hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2$ that takes from two independent variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2)$ to only one, $\hat{\mathbf{x}}$. It is known that the pdf of the sum is

$$f_{\hat{\mathbf{x}}}(x) = \int_{-\infty}^{\infty} dx_1 \, f_{\hat{\mathbf{x}}_1}(x_1) \, f_{\hat{\mathbf{x}}_2}(x - x_1). \tag{3.12}$$

Imagine, for example, that $\hat{\mathbf{x}}_1$ follows a gamma distribution $\hat{\mathbf{\Gamma}}(\alpha_1)$, and $\hat{\mathbf{x}}_2$ follows a gamma distribution $\hat{\mathbf{\Gamma}}(\alpha_2)$. The pdf of the sum is

$$\begin{aligned} f_{\hat{\mathbf{x}}}(x) &= \int_0^x dx_1 \frac{x^{\alpha_1 - 1} e^{-x_1}}{\Gamma(\alpha_1)} \frac{(x - x_1)^{\alpha_2 - 1} e^{-(x - x_1)}}{\Gamma(\alpha_2)} \\ &= \frac{x^{\alpha_1 + \alpha_2 - 1} e^{-x}}{\Gamma(\alpha_1 + \alpha_2)} \end{aligned} \tag{3.13}$$

for $x \geq 0$. This is a $\hat{\mathbf{\Gamma}}(\alpha_1 + \alpha_2)$ distribution! This is called the α-addition property. A simple iteration of this rule tells us that, to generate a gamma distribution whose index α is an integer number, all we have to do is to add up α independent $\hat{\mathbf{\Gamma}}(1)$ variables, that is, variables following an exponential distribution generated, as we know, as $-\log(u)$, with u obtained from a uniform $\hat{\mathbf{U}}(0, 1)$ distribution. A simple program is as follows:

```
double precision function ran_gamma(alpha)
 implicit none
 integer :: i,alpha
```

```
  ran_gamma=0.0d0
  do i=1,alpha
    ran_gamma=ran_gamma-dlog(ran_u())
  enddo
```

```
end function ran_gamma
```

If α is not an integer number, this α-additivity property allows us to reduce the problem to the case $\alpha \in (0, 1)$. For this case, particularly appropriate are the rejection methods, which will be the topic of the next sections (see also problem 9).

Other transformations lead also to interesting results. For instance, take two independent random variables $\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2$ with cumulative distribution functions $F_{\hat{\mathbf{x}}_1}(x_1)$, $F_{\hat{\mathbf{x}}_2}(x_2)$, and let us define a new random variable $\hat{\mathbf{z}}$ as

$$\hat{\mathbf{z}} = \max(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2). \tag{3.14}$$

The cdf of $\hat{\mathbf{z}}$ is defined as $F_{\hat{\mathbf{z}}}(z) = P(\hat{\mathbf{z}} \leq z)$, but $\hat{\mathbf{z}} \leq z$ requires $\hat{\mathbf{x}}_1 \leq z$ and $\hat{\mathbf{x}}_2 \leq z$. As $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ are independent variables, we have

$$P(\hat{\mathbf{z}} \leq z) = P(\hat{\mathbf{x}}_1 \leq z, \hat{\mathbf{x}}_2 \leq z) = P(\hat{\mathbf{x}}_1 \leq z)P(\hat{\mathbf{x}}_2 \leq z) \tag{3.15}$$

or

$$F_{\hat{\mathbf{z}}}(z) = F_{\hat{\mathbf{x}}_1}(z)F_{\hat{\mathbf{x}}_2}(z). \tag{3.16}$$

Take, for instance, the case where $F_{\hat{\mathbf{x}}_1}(x) = x^n$, $F_{\hat{\mathbf{x}}_2}(x) = x^m$ for $x \in (0, 1)$. Then $\hat{\mathbf{z}}$ has a cdf $F_{\hat{\mathbf{z}}}(z) = z^{n+m}$, a sort of α-additivity property. Iterating this procedure, we can see that taking $\hat{\mathbf{z}} = \max(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ where $\hat{\mathbf{x}}_i, i = 1, \ldots, n$ are independent $\hat{\mathbf{U}}(0, 1)$ uniform variables with $F_{\hat{\mathbf{x}}_i}(x) = x$, then leads to

$$F_{\hat{\mathbf{z}}}(z) = z^n, \quad z \in [0, 1] \tag{3.17}$$

and the corresponding pdf is

$$f_{\hat{\mathbf{z}}}(z) = nz^{n-1}. \tag{3.18}$$

Hence, a generation based on $\hat{\mathbf{z}} = \max(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ is an alternative to the formula $\hat{\mathbf{z}} = \hat{\mathbf{u}}^{1/n}$, that can be derived by a direct application of the inversion of the cdf. If n is not too a large number, it is faster to compute the maximum of n independent uniform random numbers than to exponentiate u to $1/n$ because this is a slow operation.

The same ideas can be used with discrete distributions. For instance, if $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$ are two discrete distributions taking the value $i = 0, 1, 2, \ldots$ with probability $p_i^{(1)}$ and $p_i^{(2)}$, respectively, then $\hat{\mathbf{x}} = \hat{\mathbf{x}}_1 + \hat{\mathbf{x}}_2$ adopts the value i with probability

$$p_i = \sum_{k=0}^{i} p_k^{(1)} p_{i-k}^{(2)}. \tag{3.19}$$

For example, if $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ both follow a geometrical distribution with respective probabilities p_1 and p_2, that is, $p_i^{(1)} = p_1(1 - p_1)^i$ and $p_i^{(2)} = p_2(1 - p_2)^i$, then

$$p_i = \frac{p_1 p_2}{p_2 - p_1}((1 - p_1)^{i+1} - (1 - p_2)^{i+1}). \tag{3.20}$$

It is the time now to discuss another algorithm for the generation of the binomial distribution $\hat{\mathbf{B}}(N, p)$. We have shown how to implement the inversion of the cumulative function in Section 2.4. We use here the fact that a binomial distribution appears as the sum of Bernoulli distributions. Therefore, we could generate N repetitions of an event with probability p and add up the positive results. The algorithm would be as follows:

```
integer function iran_binomial2(N,p)
 implicit none
 integer , intend (in) :: N
 double precision, intend(in) :: p
 double precision ran_u
  iran_binomial2=0
  do i=1,N
    if (ran_u() < p) iran_binomial2=iran_binomial2+1
  enddo
end function iran_binomial2
```

This algorithm implies the generation of N uniform random numbers, while the one of Section 2.4 requires several mathematical operations. The balance is that the algorithm just presented is more efficient for large values of N.

The Poisson distribution can be generated using an interesting *ad hoc* method. We have shown, when discussing the exponential distribution in Section 1.3.6, a relation between the Poisson and the exponential distribution. This relation says that, if $\hat{\mathbf{t}}$ follows an exponential distribution with pdf

$$f_{\hat{\mathbf{t}}}(t) = \lambda e^{-\lambda t} \tag{3.21}$$

then the number of events occurring in a time interval $(0, 1)$ follows a Poisson $\hat{\mathbf{P}}(\lambda)$ distribution. As the values t_i of this random variable can be obtained from

$$t_i = \frac{-1}{\lambda} \log u_i \tag{3.22}$$

u_i being the independent values of a $\hat{\mathbf{U}}(0, 1)$ distribution, we need to count the events until the total time exceeds $t = 1$. Hence, if m are integer values distributed according to the Poisson distribution $\hat{\mathbf{P}}(\lambda)$, then the times t_i verify

$$\sum_{i=0}^{m} t_i \leq 1 < \sum_{i=0}^{m+1} t_i \tag{3.23}$$

which, using (3.22), can be written as

$$-\frac{1}{\lambda} \sum_{i=0}^{m} \log u_i \leq 1 < -\frac{1}{\lambda} \sum_{i=0}^{m+1} \log u_i \tag{3.24}$$

or

$$\prod_{i=0}^{m} u_i \geq e^{-\lambda} > \prod_{i=0}^{m+1} u_i \tag{3.25}$$

as the condition to verify by the number m. This can be implemented by the following program:

```
integer function iran_poisson(lambda)
 implicit none
 double precision, intend (in) :: lambda
 double precision :: u,ran_u
```

```
u=ran_u()
iran_poisson=0
do while(u > exp(-lambda))
  u=u*ran_u()
  iran_poisson=iran_poisson+1
enddo
```

```
end function iran_poisson
```

The examples could continue forever, but our intention has been just to show that a little bit of thinking combined with some intuition can simplify the problem of the generation of random numbers for many complicated distributions. We will just explain a last trick that can be useful in a variety of occasions.

3.3 Combination of Variables

Imagine that the pdf $f_{\hat{\mathbf{x}}}(x)$ of the random variable $\hat{\mathbf{x}}$ whose values we want to generate can be split as

$$f_{\hat{\mathbf{x}}}(x) = \sum_i p_i \, f_i(x) \tag{3.26}$$

where the functions $f_i(x)$ can also be considered as the pdf of some random variables $\hat{\mathbf{x}}_i$. All we need for that is nonnegativity $f_i(x) \geq 0$ and normalization $\int dx f_i(x) = 1$. From the latter, we can derive easily

$$\sum_i p_i = 1. \tag{3.27}$$

If, furthermore, $p_i > 0 \; \forall i$, we can interpret p_i as the different probabilities of the outcomes of a discrete random variable $\hat{\mathbf{z}}$ that takes integer values, with pdf

$$f_{\hat{\mathbf{z}}}(z) = \sum_i p_i \delta(z - i). \tag{3.28}$$

The generation of a value for $\hat{\mathbf{x}}$ proceeds in two steps: first, we generate a value of $\hat{\mathbf{z}}$, say $\hat{\mathbf{z}} = i$, and then we generate a value of the random variable distributed according to $f_i(x)$. Using (1.135), the resulting pdf of this two-step process

$$f_{\hat{\mathbf{x}}}(x) = \sum_i \text{Prob}(\hat{\mathbf{z}} = i) f_{\hat{x}}(x|\hat{\mathbf{z}} = i) = \sum_i p_i \, f_i(x) \tag{3.29}$$

is the pdf $f_{\hat{\mathbf{x}}}(x)$ we wanted to sample.

Let us see an example. Imagine we want to generate the values of a random variable $\hat{\mathbf{x}}$ with pdf

$$f_{\hat{\mathbf{x}}}(x) = \frac{5}{6}(1 + x^4), \qquad x \in (0, 1) \tag{3.30}$$

which we write in the form

$$f_{\hat{\mathbf{x}}}(x) = \frac{5}{6} 1 + \frac{1}{6}(5x^4) \equiv p_1 \, f_1(x) + p_2 \, f_2(x). \tag{3.31}$$

$f_1(x) = 1$ is the pdf of a uniform $\hat{\mathbf{U}}(0,1)$ variable. This occurs with probability $p_1 = 5/6$. Using the inverse cdf, $f_2(x) = 5x^4$ can be generated using $x = u^{1/5}$, with u being a value from a $\hat{\mathbf{U}}(0,1)$ variable. This second distribution should be used with probability $p_2 = 1/6$. The algorithm can be programmed as follows:

```
double precision function ran_f()
 implicit none
 double precision ran_u

  if (ran_u() > 1.0d0/6.0d0) then
    ran_f=ran_u()
  else
    ran_f=ran_u()**0.2d0
  endif

end function ran_f
```

Some reflexion shows that this program is equivalent to the following:

```
double precision function ran_f()
 implicit none
 double precision ran_u

  ran_f=ran_u()
  if (ran_u() < 1.0d0/6.0d0) ran_f=ran_f**0.2d0

end function ran_f
```

Sometimes, the splitting given by (3.26) is not so obvious. Remember, it is essential that $p_i > 0$. Take, for instance, the pdf

$$f_{\hat{\mathbf{x}}}(x) = \frac{4}{7}(1 + 3x^2 - x^3), \qquad x \in (0,1) \tag{3.32}$$

which we split as

$$f_{\hat{\mathbf{x}}}(x) = \frac{1}{7}(4(1-x)^3) + \frac{6}{7}(2x) \equiv p_1\ f_1(x) + p_2\ f_2(x) \tag{3.33}$$

so $p_1 = 1/7,\ p_2 = 6/7,\ f_1(x) = 4(1-x)^3,\ f_2(x) = 2x, f_1(x)$, and $f_2(x)$ can be sampled easily as $1 - (1-u)^{1/4} \equiv 1 - u^{1/4}$ and $u^{1/2}$, respectively. The program is as follows:

```
double precision function ran_f()
 implicit none
 double precision ran_u

  ran_f=sqrt(ran_u())
  if (ran_u() < 1.0d0/7.0d0) ran_f=1.0d0-dsqrt(ran_f)

end function ran_f
```

Here comes an interesting example. Imagine we need to sample the pdf

$$f_{\hat{\mathbf{x}}}(x) = e^{-x-1/4} I_0(\sqrt{x}), \quad x \geq 0. \tag{3.34}$$

We expand in a Taylor series the modified Bessel function I_0 to obtain

$$f_{\hat{\mathbf{x}}}(x) = e^{-x-1/4} \sum_{i=0}^{\infty} \frac{\left(\frac{\sqrt{x}}{2}\right)^{2i}}{(i!)^2} = \sum_{i=0}^{\infty} e^{-1/4} \frac{\left(\frac{1}{4}\right)^i}{i!} e^{-x} \frac{x^i}{i!} \tag{3.35}$$

which is of the form (3.26) with $p_i = e^{-1/4} \frac{\left(\frac{1}{4}\right)^i}{i!}$, a Poisson distribution of parameter $\lambda = 1/4$, and $f_i(x) = e^{-x} \frac{x^i}{i!}$, a gamma distribution with integer parameter $\alpha = i + 1$. All we need to do to generate the distribution $f_{\hat{\mathbf{x}}}(x)$ is to get an integer number i from a Poisson distribution of parameter $1/4$ and then generate a random number according to the gamma distribution with parameter $i + 1$, which is equivalent to adding $i + 1$ random numbers distributed according to an exponential distribution. A possible program would be as follows:

```
double precision function ran_I0()
 implicit none
 double precision ran_gamma
 integer k,iran_poisson

  k=iran_poisson(0.25d0)+1
  ran_I0=ran_gamma(k)

end function ran_I0
```

3.3.1 A Rejection Method

The method of combination of variables has an interesting twist. It suffices to consider the case of two variables for which (3.29) reduces to

$$\begin{aligned} f_{\hat{\mathbf{x}}}(x) &= \text{Prob}(\hat{\mathbf{z}} = 1) f(x|\hat{\mathbf{z}} = 1) + \text{Prob}(\hat{\mathbf{z}} = 2) f(x|\hat{\mathbf{z}} = 2) \\ &= p_1 \, f_{\hat{\mathbf{x}}_1}(x) + p_2 \, f_{\hat{\mathbf{x}}_2}(x). \end{aligned} \tag{3.36}$$

Imagine now that the random variable $\hat{\mathbf{x}}$ can be easily generated, but not so $\hat{\mathbf{x}}_1$ (we do not care much about $\hat{\mathbf{x}}_2$ in this process). If we generated a value x from $f_{\hat{\mathbf{x}}}(x)$ using the method of combination of variables, we know that it could have come from the generation of $\hat{\mathbf{x}}_1$ or of $\hat{\mathbf{x}}_2$. Let us be more precise using Bayes theorem. Given a value x of the random variable $\hat{\mathbf{x}}$, the probability $P(\hat{\mathbf{z}} = 1|x)$ that it comes from the variable $\hat{\mathbf{x}}_1$ is

$$P(\hat{\mathbf{z}} = 1|x) = \frac{f(x|\hat{\mathbf{z}} = 1) P(\hat{\mathbf{z}} = 1)}{f_{\hat{\mathbf{x}}}(x)} = \frac{f_{\hat{\mathbf{x}}_1}(x) p_1}{f_{\hat{\mathbf{x}}}(x)}. \tag{3.37}$$

The idea now is to generate a value of x from $f_{\hat{\mathbf{x}}}(x)$ (which is supposed to be easy) and keep this value with probability $P(\hat{\mathbf{z}} = 1|x)$. In this way, we keep it only when x

corresponds to $\hat{\mathbf{x}}_1$ and, effectively, we generate values of $\hat{\mathbf{x}}_1$ (which was supposed to be difficult). The price to pay is that not all values of x are valid, and we only keep a fraction of them. If we do not keep a value, we need to repeat the process until we generate a valid number.

Let us give an example. Let us consider $f_{\hat{\mathbf{x}}}(x) = 2x$. This can be generated as $x = \sqrt{u}$; $f_{\hat{\mathbf{x}}_1}(x) = 6x(1-x)$. This is the difficult one to generate as the cumulative function is $F_{\hat{\mathbf{x}}_1} = 3x^2 - 2x^3$ and the inverse function of a third-degree polynomial is not so easy to compute. We write then the identity $f_{\hat{\mathbf{x}}}(x) = p_1 f_{\hat{\mathbf{x}}_1}(x) + p_2 f_{\hat{\mathbf{x}}_2}(x)$ in the form

$$2x = \frac{1}{3}6x(1-x) + \frac{2}{3}3x^2 \tag{3.38}$$

which identifies $P(\hat{\mathbf{z}} = 1) = p_1 = \frac{1}{3}$; $f_{\hat{\mathbf{x}}_2} = 3x^2$. This is also easy to generate, but we do not care. Bayes theorem then tells us

$$P(\hat{\mathbf{z}} = 1|x) = \frac{f_{\hat{\mathbf{x}}_1}(x)p_1}{f_{\hat{\mathbf{x}}}(x)} = \frac{6x(1-x)\frac{1}{3}}{2x} = 1 - x. \tag{3.39}$$

Then, in order to generate $f_{\hat{\mathbf{x}}_1}(x) = 6x(1-x)$, we generate a value x according to $f_{\hat{\mathbf{x}}}(x) = 2x$ using $x = \sqrt{u}$ and then accept it with probability $1 - x$. The acceptance is a Bernoulli process (either we accept or not) and we do it by comparing the acceptance probability $1 - x$ with a second uniform random number v: if $v < 1 - x$, we accept it, or equivalently, if $x < 1 - v \equiv v$, as $1 - v$ and v are both uniform random numbers. In other words, if $x > v$, we do not accept (we reject) the proposed number x, as it is not representative of $f_{\hat{\mathbf{x}}_1}(x)$. When we reject, we need to repeat the process again until we accept the proposed number. The final algorithm to sample $f_{\hat{\mathbf{x}}_1}(x) = 6x(1-x)$ is as follows:

```
double precision function ran_f()
 implicit none
 double precision ran_u

  do
    ran_f=sqrt(ran_u())
    if (ran_f < ran_u()) exit
  enddo

end function ran_f
```

This is a very simple algorithm. Its main problem is that the average acceptance probability is $p_1 = 1/3$, so we discard two out of every three random numbers generated. This is an example of a *rejection* algorithm, where not all proposed values are kept. General rejection algorithms will be analyzed in Section 3.6, and they constitute one of the most powerful algorithms to generate very general probability distributions.

3.4 Multidimensional Distributions

It is not easy to extend the previous techniques to the case of multidimensional distributions. The problem is to generate the values of a vector or random variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \dots, \hat{\mathbf{x}}_n)$ with a joint probability density function $f_{\hat{\mathbf{x}}_0,\dots,\hat{\mathbf{x}}_n}(x_1, \dots, x_n)$. The equivalent method to the inversion formula $\hat{\mathbf{x}} = F^{-1}_{\hat{\mathbf{x}}}(u)$ for one variable is based on the splitting of the joint pdf in conditional probabilities:

$$f_{\hat{\mathbf{x}}_0,\dots,\hat{\mathbf{x}}_n}(x_1, \dots, x_n) = f_{\hat{\mathbf{x}}_1}(x_1) f_{\hat{\mathbf{x}}_2}(x_2|x_1) \cdots f_{\hat{\mathbf{x}}_n}(x_n|x_1, \dots, x_{n-1}). \tag{3.40}$$

So, what we do is to generate first an independent value of $\hat{\mathbf{x}}_1$ according to $f_{\hat{\mathbf{x}}_1}(x_1)$; given this value x_1, we generate a value for $\hat{\mathbf{x}}_2$ based on the conditional distribution $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$; given x_1 and x_2, we generate a value for $\hat{\mathbf{x}}_3$ based on $f_{\hat{\mathbf{x}}_3}(x_3|x_1, x_2)$, and so on. The process starts by generating a value of the n-dimensional vector $(u_1, \dots, u_n)$ of independent uniform $\hat{\mathbf{U}}(0, 1)$ variables, and then find $(x_1, \dots, x_n)$ as the solution of the following system of equations:

$$\begin{aligned}
F_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{x_1} dx_1' \, f_{\hat{\mathbf{x}}_1}(x_1') = u_1, \\
F_{\hat{\mathbf{x}}_2}(x_2|x_1) &= \int_{-\infty}^{x_2} dx_2' \, f_{\hat{\mathbf{x}}_2}(x_2'|x_1) = u_2, \\
&\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \\
F_{\hat{\mathbf{x}}_n}(x_n|x_1, \dots, x_{n-1}) &= \int_{-\infty}^{x_n} dx_n' \, f_{\hat{\mathbf{x}}_n}(x_n'|x_1, \dots, x_{n-1}) = u_n.
\end{aligned} \tag{3.41}$$

Neither the calculation of n cumulative distribution functions nor the solution of this system of equations is easy to carry out in most cases of interest. Still we can give some examples of the application of this procedure. We consider $n = 2$, and let the random variables $\hat{\mathbf{x}}_1$, $\hat{\mathbf{x}}_2$ with joint pdf

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = \begin{cases} 2x_1 x_2 e^{-x_1(1+x_2^2)}, & \text{if } x_1 \geq 0, x_2 \geq 0, \\ 0, & \text{else.} \end{cases} \tag{3.42}$$

We first compute $f_{\hat{\mathbf{x}}_1}(x_1)$ and the corresponding cumulative function $F_{\hat{\mathbf{x}}_1}(x_1)$:

$$f_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{\infty} dx_2 \, f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = e^{-x_1}, \; x_1 \geq 0, \tag{3.43}$$

$$F_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{x_1} dx_1' \, f_{\hat{\mathbf{x}}_1}(x_1') = 1 - e^{-x_1} \tag{3.44}$$

which is an exponential distribution, generated by $x_1 = -\log(u_1)$. The conditional probability $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$ is

$$f_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2)}{f_{\hat{\mathbf{x}}_1}(x_1)} = 2x_1 x_2 e^{-x_1 x_2^2} \tag{3.45}$$

and the cumulative conditional probability is

$$F_{\hat{\mathbf{x}}_2}(x_2|x_1) = \int_{-\infty}^{x_2} dx_2' \, f_{\hat{\mathbf{x}}_2}(x_2'|x_1) = 1 - e^{-x_1 x_2^2}. \tag{3.46}$$

The solution of $F_{\hat{\mathbf{x}}_2}(x_2|x_1) = u_2$ is $x_2 = \sqrt{-\frac{\log(1-u_2)}{x_1}} \equiv \sqrt{-\frac{\log u_2}{x_1}}$. We implement this in the following program listing:

```
double precision function ran_f()
 implicit none
 double precision :: ran_u
 double precision, dimension(2):: ran_f
```

```
  ran_f(1)=-log(ran_u())
  ran_f(2)=sqrt(-log(ran_u())/ran_f(1))
```

```
end function ran_f
```

Of course, one can interchange the role of $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ and compute first $f_{\hat{\mathbf{x}}_2}(x_2)$ and then $f_{\hat{\mathbf{x}}_1}(x_1|x_2)$. In this case, this would give

$$f_{\hat{\mathbf{x}}_2}(x_2) = \int_{-\infty}^{\infty} dx_1\, f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = \frac{2x_2}{(1+x_2^2)^2},\ x_2 \geq 0, \tag{3.47}$$

$$F_{\hat{\mathbf{x}}_2}(x_2) = \int_{-\infty}^{x_2} dx_2'\, f_{\hat{\mathbf{x}}_2}(x_2') = \frac{x_2^2}{1+x_2^2}. \tag{3.48}$$

Hence, the solution of $F_{\hat{\mathbf{x}}_2}(x_2) = u$ is $x_2 = \sqrt{\frac{1-u}{u}}$. For $\hat{\mathbf{x}}_1$, we compute

$$f_{\hat{\mathbf{x}}_1}(x_1|x_2) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2)}{f_{\hat{\mathbf{x}}_2}(x_2)} = (1+x_2^2)^2 x_1 e^{-x_1(1+x_2^2)}. \tag{3.49}$$

To make things clearer, let us define $a = 1 + x_2^2$. The distribution is $f_{\hat{\mathbf{x}}_1}(x_1|x_2) = a^2 x_1 e^{-a x_1}$. So $a x_1$ is a $\hat{\mathbf{\Gamma}}(2)$ variable which we already know can be generated as the sum of two independent exponential variables: $a x_1 = -\log(u_1) - \log(u_2)$. As $a = 1 + x_2^2 = 1/u$, the final algorithm can be written as follows:

```
double precision function ran_f()
 implicit none
 double precision :: u,ran_u
 double precision, dimension(2):: ran_f
```

```
  u=ran_u()
  ran_f(2)=dsqrt((1.0d0-u)/u)
  ran_f(1)=u*(-log(ran_u())-log(ran_u()))
```

```
end function ran_f
```

In this example, it was possible to find first $f_{\hat{\mathbf{x}}_1}(x_1)$ and then $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$, or to reverse the order and find first $f_{\hat{\mathbf{x}}_2}(x_2)$ and then $f_{\hat{\mathbf{x}}_1}(x_1|x_2)$. Sometimes, the order in which we consider the variables does matter. Consider the distribution

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = \begin{cases} C\frac{x_1}{x_1^2+x_2^2+1}, & 0 \leq x_1 \leq 1, x_2 \geq 0, \\ 0, & \text{else.} \end{cases} \tag{3.50}$$

The normalization constant is $C = \frac{2(\sqrt{2}+1)}{\pi}$. We first compute $f_{\hat{\mathbf{x}}_1}$:

$$f_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{\infty} dx_1\, f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = \left(\sqrt{2}+1\right) \frac{x_1}{\sqrt{1+x_1^2}} \tag{3.51}$$

and the cumulative distribution

$$\begin{aligned} F_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{x_1} dx_1'\, f_{\hat{\mathbf{x}}_1}(x_1') \\ &= \int_{0}^{x_1} dx_1' \left(\sqrt{2}+1\right) \frac{x_1'}{\sqrt{1+x_1'^2}} = \left(\sqrt{2}+1\right)\left(\sqrt{1+x_1^2}-1\right) \end{aligned} \tag{3.52}$$

and the solution of $F_{\hat{\mathbf{x}}_1}(x_1) = u_1$ is $x_1 = \left(\frac{u_1}{\sqrt{2}+1}+1\right)^2 - 1$. Once x_1 has been found, we find $f_{\hat{\mathbf{x}}_2}(x_2|x_1)$. Indeed, it is not necessary to actually compute the ratio $f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2)/f_{\hat{\mathbf{x}}_1}(x_1)$. All we effectively need to do is to consider that x_1 is just a known constant. The distribution of $\hat{\mathbf{x}}_2$, given that $\hat{\mathbf{x}}_1$ takes the value x_1, is

$$f_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{2A}{\pi}\frac{1}{x_2^2 + A^2}, \qquad x_2 \geq 0 \tag{3.53}$$

with $A^2 = x_1^2 + 1$. The cumulative function is $F_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{2}{\pi}\arctan\left(\frac{x_2}{A}\right)$, and the solution of $F_{\hat{\mathbf{x}}_2}(x_2|x_1) = u_2$ is $x_2 = A\tan\left(\frac{\pi}{2}u_2\right)$. If, on the other hand, we compute first $f_{\hat{\mathbf{x}}_2}$, we obtain

$$f_{\hat{\mathbf{x}}_2}(x_2) = \int_{-\infty}^{\infty} dx_1' dx_1\; f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = \frac{\sqrt{2}+1}{2}\log\left[\frac{2+x_2^2}{1+x_2^2}\right] \tag{3.54}$$

and the cumulative distribution is

$$\begin{aligned} &F_{\hat{\mathbf{x}}_2}(x_2) \\ &= \frac{\sqrt{2}+1}{2}\left(-2\arctan(x_2) + 2\sqrt{2}\arctan\left(\frac{x_2}{\sqrt{2}}\right) + x_2\log\left[\frac{2+x_2^2}{1+x_2^2}\right]\right), \end{aligned} \tag{3.55}$$

and it is not easy to solve $F_{\hat{\mathbf{x}}_2}(x_2) = u_2$ and find x_2.

As a final example, let us consider an innocent-looking distribution again in two dimensions:

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) = \begin{cases} \frac{1}{\pi}, & \text{if } x_1^2 + x_2^2 \leq 1, \\ 0, & \text{else.} \end{cases} \tag{3.56}$$

This is a uniform distribution in a circle of radius 1. If we follow our general setup, we first compute $f_{\hat{\mathbf{x}}_1}(x_1)$ and the corresponding cumulative function $F_{\hat{\mathbf{x}}_1}(x_1)$:

$$\begin{aligned} f_{\hat{\mathbf{x}}_1}(x_1) &= \int_{-\infty}^{\infty} dx_1' dx_2\, f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1, x_2) \\ &= \int_{-\sqrt{1-x_1^2}}^{+\sqrt{1-x_1^2}} dx_2 \frac{1}{\pi} = \frac{2}{\pi}\sqrt{1-x_1^2}, \text{ for } -1 \leq x_1 \leq 1, \end{aligned} \tag{3.57}$$

$$F_{\hat{\mathbf{x}}_1}(x_1) = \int_{-\infty}^{x_1} dx_1' f_{\hat{\mathbf{x}}_1}(x_1') = \int_{-\infty}^{x_1} dx_1' \frac{2}{\pi}\sqrt{1-x_1'^2}$$
$$= \frac{1}{2} + \frac{1}{\pi}\left(x_1\sqrt{1-x_1^2} + \arcsin x_1\right). \tag{3.58}$$

Solving $F_{\hat{\mathbf{x}}_1}(x_1) = u_1$ might require some work (maybe use Newton–Raphson algorithm). However, once x_1 has been found, the pdf for $\hat{\mathbf{x}}_2$ is not difficult:

$$f_{\hat{\mathbf{x}}_2}(x_2|x_1) = \frac{f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2}(x_1,x_2)}{f_{\hat{\mathbf{x}}_1}(x_1)} = \frac{1}{2\sqrt{1-x_1^2}}, \text{ for } |x_2| \le \sqrt{1-x_1^2} \tag{3.59}$$

which is nothing but a uniform distribution in the interval $(-\sqrt{1-x_1^2}, \sqrt{1-x_1^2})$ obtained by $x_2 = (2u_2 - 1)\sqrt{1-x_1^2}$. We will not write the corresponding program, as there are more efficient methods to generate this distribution. For example, we might go over to polar coordinates, as given by (3.5), to find that the joint pdf for $(\hat{\mathbf{r}}, \hat{\boldsymbol{\theta}})$ is particularly simple (if we do not forget the Jacobian of the transformation):

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\theta}}}(r,\theta) = r f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2} = r\frac{1}{\pi}. \qquad \text{if } r \le 1. \tag{3.60}$$

This is now the product of two independent distributions: $f_{\hat{\mathbf{r}},\hat{\boldsymbol{\theta}}}(r,\theta) = f_{\hat{\mathbf{r}}}(r) f_{\hat{\boldsymbol{\theta}}}(\theta)$, with $f_{\hat{\mathbf{r}}}(r) = 2r$ and $f_{\hat{\boldsymbol{\theta}}}(\theta) = \frac{1}{2\pi}$. Hence, $\hat{\boldsymbol{\theta}}$ is uniformly distributed in $(0, 2\pi)$ or $\theta = 2\pi u_1$, and r is obtained from the solution of $F_{\hat{\mathbf{r}}}(r) = r^2 = u_2$ or $r = \sqrt{u_2}$. The algorithm can be written as follows:

```
function ran_f()
  implicit none
  double precision,dimension(2)::ran_f
  double precision :: theta,r,ran_u,pi
  data /pi/ 3.14159265359
  theta=2.0d0*pi*ran_u()
  r=dsqrt(ran_u())
  ran_f(1)=r*cos(theta)
  ran_f(2)=r*sin(theta)
end function ran_f
```

The same idea can be applied to the generation of points distributed uniformly in a three-dimensional sphere of radius 1. The pdf is

$$f_{\hat{\mathbf{x}}_1,\hat{\mathbf{x}}_2,\hat{\mathbf{x}}_3}(x_1,x_2,x_3) = \begin{cases} \frac{3}{4\pi}, & \text{if } x_1^2 + x_2^2 + x_3^2 \le 1, \\ 0, & \text{else.} \end{cases} \tag{3.61}$$

We change to spherical coordinates (r, ϕ, θ), with $\phi \in (0, \pi)$, $\theta \in (0, 2\pi)$ defined by

$$x_1 = r\sin\phi\sin\theta,$$
$$x_2 = r\cos\phi\sin\theta,$$
$$x_3 = r\cos\theta. \tag{3.62}$$

The Jacobian is $J = r^2\sin\phi$, and the pdf in polar coordinates is

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\phi}},\hat{\boldsymbol{\theta}}}(r,\phi,\theta) = \frac{3}{4\pi}r^2\sin\phi \tag{3.63}$$

which can be written as product of independent distributions:

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\phi}},\hat{\theta}}(r,\phi,\theta) = f_{\hat{\mathbf{r}}}(r) f_{\hat{\boldsymbol{\phi}}}(\phi) f_{\hat{\theta}}(\theta) = 3r^2 \times \frac{1}{2}\sin(\phi) \times \frac{1}{2\pi}. \tag{3.64}$$

It is simple now to obtain (r,ϕ,θ) from the values (u_1,u_2,u_3) independently taken from a uniform $\hat{\mathbf{U}}(0,1)$ random variable. Considering that $F_{\hat{r}}(r) = r^3$, $F_{\hat{\phi}}(\phi) = \frac{1}{2}(1+\cos\phi)$, $f_{\hat{\theta}}(\theta) = \theta/2\pi$ we obtain:

$$r = u_1^{1/3}, \tag{3.65}$$

$$\phi = \arccos(2u_2 - 1) \Rightarrow \cos\phi = 2u_2 - 1, \sin\phi = 2\sqrt{u_2(1-u_2)}, \tag{3.66}$$

$$\theta = 2\pi u_3. \tag{3.67}$$

However, the change to spherical coordinates is not very useful for an n-dimensional sphere for $n > 3$. The pdf is

$$f_{\hat{\mathbf{x}}_0,\ldots,\hat{\mathbf{x}}_n}(x_1,\ldots,x_n) = \begin{cases} 1/V_n, & x_1^2 + \cdots + x_n^2 \le 1 \\ 0, & \text{otherwise,} \end{cases} \tag{3.68}$$

with $V_n = \frac{\pi^{n/2}}{\Gamma(n/2+1)}$, the volume of the n-dimensional sphere of radius 1. In this case, one can use a trick that can be applied to any multidimensional distribution which depends only on the modulus $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n) = f(r)$ with $r^2 = x_1^2 + \cdots + x_n^2$. In the previous example, it is

$$f(r) = \begin{cases} 1/V_n, & r \le 1 \\ 0, & \text{otherwise.} \end{cases} \tag{3.69}$$

Using the change to spherical coordinates, modulus r and $n-1$ angles $(\phi_1,\ldots,\phi_{n-1})$, we can write the pdf in these coordinates as

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\phi}}_1,\ldots,\hat{\boldsymbol{\phi}}_{n-1}}(r,\phi_1,\ldots,\phi_{n-1}) = J\left(\frac{r,\phi_1,\ldots,\phi_{n-1}}{x_1,\ldots,x_n}\right) f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n). \tag{3.70}$$

As the Jacobian is $nV_n r^{n-1}\Omega(\phi_1,\ldots,\phi_{n-1})$ with $\Omega(\phi_1,\ldots,\phi_{n-1})$ a function of the angles, and $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n) = f(r)$, the pdf can be split as

$$f_{\hat{\mathbf{r}},\hat{\boldsymbol{\phi}}_1,\ldots,\hat{\boldsymbol{\phi}}_{n-1}}(r,\phi_1,\ldots,\phi_{n-1}) = \Omega(\phi_1,\ldots,\phi_{n-1}) n V_n r^{n-1} f(r) \tag{3.71}$$

$$\equiv f_{\hat{\mathbf{r}}}(r) f_{\hat{\boldsymbol{\phi}}_1,\ldots,\hat{\boldsymbol{\phi}}_{n-1}}(\phi_1,\ldots,\phi_{n-1}) \tag{3.72}$$

with $f_{\hat{\mathbf{r}}}(r) = Cr^{n-1}f(r)$, and C a normalization constant required to make $f_{\hat{\mathbf{r}}}(r)$ a true pdf verifying $\int_0^\infty dr f_{\hat{\mathbf{r}}}(r) = 1$. All we have to do now is to sample this one-dimensional distribution $f_{\hat{\mathbf{r}}}(r)$ to obtain the modulus r. Concerning the generation of the angles $(\phi_1,\ldots,\phi_{n-1})$, we notice that their pdf $f_{\hat{\boldsymbol{\phi}}_1,\ldots,\hat{\boldsymbol{\phi}}_{n-1}}(\phi_1,\ldots,\phi_{n-1}) = A\Omega(\phi_1,\ldots,\phi_{n-1})$ (with A another normalization constant) is the same for all distributions of the form $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n) = f(r)$. Then, if we can find one pdf $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n)$ which depends only on the modulus, and generate points $(x_1,\ldots,x_n)$ according to this distribution, then the obtained angles can be used in

any other pdf of the same form. One pdf that depends only on the modulus r and can be sampled easily is the product of Gaussian pdfs:

$$f_{\hat{\mathbf{x}}_0,\ldots,\hat{\mathbf{x}}_n}(x_1,\ldots,x_n)=\prod_{k=1}^{n}\frac{1}{\sqrt{2\pi}}e^{-x_i^2/2}=\frac{1}{(2\pi)^{n/2}}e^{-r^2/2}. \tag{3.73}$$

Once the x_i's have been generated using this distribution, all that remains to be done is to correct for the modulus by generating a value of r with the correct distribution $f_{\hat{\mathbf{r}}}(r)$.

In summary, the procedure to generate an n-dimensional distribution which only depends on the modulus, $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n)=f(r)$, is the following:

1) Generate a value of the modulus r using the pdf $f_{\hat{\mathbf{r}}}(r)=Cr^{n-1}f(r)$, with C being a normalization constant found by imposing $\int_0^\infty dr\, f_{\hat{\mathbf{r}}}(r)=1$.
2) Generate n independent Gaussian $\hat{\mathbf{G}}(0,1)$ variables $(x_1,\ldots,x_n)$.
3) Use the modulus of (1) and the angles of (2).

This last step can be done simply by rescaling all variables $(x_1,\ldots,x_n)$ obtained in (2) such that the resulting modulus is r, that is, setting $x_i\to\frac{x_i}{\sqrt{x_1^2+\cdots+x_n^2}}r$. Let us give a computer program to implement this algorithm:

```
subroutine ran_f(x,n)
  implicit none
  integer n
  double precision (:) ::  x(n)
  double precision r,ran_g,ran_fr
```

```
do i=1,n
 x(i)=ran_g()
enddo
q=dsqrt(sum(x*x))
r=ran_fr()
x=x/q*r
```

```
end subroutine ran_f
```

In this program, the line `r=ran_fr()` produces a value of the modulus according to the distribution $f_{\hat{\mathbf{r}}}(r)$. For example, in the distribution (3.68), it is $f_{\hat{\mathbf{r}}}(r)=Cr^{n-1}$ if $r\le 1$ and the normalization constant is $C=n$. The cumulative distribution is $F_{\hat{\mathbf{r}}}(r)=r^n$, and this can be sampled by $r=u^{1/n}$, with u being the value of a uniform $\hat{\mathbf{U}}(0,1)$ distribution.

3.5 Gaussian Distribution

One of the few n-dimensional distributions that can be generated without using the rejection methods, which we will explain later, is the Gaussian distribution.

The joint pdf is (1.80), which we repeat here:

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, x_2, \ldots, x_n) = \sqrt{\frac{|\mathbf{A}|}{(2\pi)^n}} \exp\left[-\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(x_i - \mu_i)A_{ij}(x_j - \mu_j)\right] \tag{1.80}$$

where μ_i is the average of $\hat{\mathbf{x}}_i$ and the symmetric matrix $\mathbf{A}$ is the inverse of the correlation matrix $\mathbf{C}$:

$$\mu_i = \langle \hat{\mathbf{x}}_i \rangle, \tag{3.74}$$

$$C_{ij} = \langle (\hat{\mathbf{x}}_i - \mu_i)(\hat{\mathbf{x}}_j - \mu_j) \rangle. \tag{3.75}$$

We take the set of numbers μ_i and C_{ij} as the given parameters, and we want to generate n Gaussian-distributed random variables $(\hat{\mathbf{x}}_1, \hat{\mathbf{x}}_2, \ldots, \hat{\mathbf{x}}_n)$ whose averages and correlations are given by (3.74) and (3.75).

The method resembles somehow Schmidt's orthonormalization procedure, and it is based on the property that a linear combination of Gaussian random variables is also a Gaussian random variable. We start from n independent Gaussian random variables $\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_n$ of zero mean and variance 1, that is, $\langle \hat{\mathbf{z}}_i \rangle = 0$, $\langle \hat{\mathbf{z}}_i \hat{\mathbf{z}}_j \rangle = \delta_{ij}$. They are generated by our favorite Gaussian random generator (e.g., Box–Muller–Wiener). Then we start defining new random variables $\hat{\mathbf{x}}_i$ as linear combinations of the $\hat{\mathbf{z}}_i$'s in such a way that (3.74) and (3.75) are satisfied.

The first variable $\hat{\mathbf{x}}_1$ is obtained as

$$\hat{\mathbf{x}}_1 = b_{11}\hat{\mathbf{z}}_1 + \mu_1. \tag{3.76}$$

As a linear combination of Gaussian variables, $\hat{\mathbf{x}}_1$ is also Gaussian. Also, it is clear that $\langle \hat{\mathbf{x}}_1 \rangle = \mu_1$. To find b_{11}, we use $C_{11} = \langle (\hat{\mathbf{x}}_1 - \mu_1)^2 \rangle = b_{11}^2 \langle \hat{\mathbf{z}}_1^2 \rangle = b_{11}^2$ or $b_{11} = C_{11}^{1/2}$.

Next we write the second variable as

$$\hat{\mathbf{x}}_2 = b_{21}\hat{\mathbf{z}}_1 + b_{22}\hat{\mathbf{z}}_2 + \mu_2. \tag{3.77}$$

Again, it is a Gaussian random variable which, trivially, satisfies $\langle \hat{\mathbf{x}}_2 \rangle = \mu_2$. Constants b_{21} and b_{22} are determined by imposing the two conditions

$$\begin{aligned} C_{12} &= \langle (\hat{\mathbf{x}}_1 - \mu_1)(\hat{\mathbf{x}}_2 - \mu_2) \rangle = b_{11} b_{21}, \\ C_{22} &= \langle (\hat{\mathbf{x}}_2 - \mu_2)^2 \rangle = b_{21}^2 + b_{22}^2 \end{aligned} \tag{3.78}$$

which leads to

$$b_{21} = \frac{C_{12}}{b_{11}}, \tag{3.79}$$

$$b_{22} = \left(C_{22} - b_{21}^2\right)^{1/2}. \tag{3.80}$$

The process is iterated. To generate the jth random variable $\hat{\mathbf{x}}_j$, we write

$$\hat{\mathbf{x}}_j = \sum_{i=1}^{j} b_{ji}\hat{\mathbf{z}}_i + \mu_j \tag{3.81}$$

and compute constants b_{ij} by the recurrence relations

$$b_{ji} = \frac{C_{ji} - \sum_{k=1}^{i-1} b_{ik} b_{jk}}{b_{ii}}, \qquad j = 1 \ldots, n, \;\; i = 1, \ldots, j-1, \tag{3.82}$$

$$b_{jj} = \sqrt{C_{jj} - \sum_{k=1}^{j-1} b_{jk}^2}. \tag{3.83}$$

A possible program implementing this algorithm is as follows:

```
subroutine generab(b,c,n)
implicit none
double precision (:,:) :: c,b
double precision z
integer i,j,k,n
do j=1,n
 do i=1,j
   z=c(j,i)
   do k=1,i-1
     z=z-b(j,k)*b(i,k)
   enddo
   if (i < j) then
     b(j,i)=z/b(i,i)
   else
     b(j,j)=sqrt(z)
   endif
 enddo
enddo
end subroutine generab

function ran_gn(n,mu,b)
implicit none
double precision (:) :: mu
double precision (:,:) :: b
double precision,dimension(n)::ran_gn
double precision,dimension(n)::z
integer i,j
```

```
do j=1,n
 z(j)=ran_g()
 ran_gn(j)=mu(j)
 do i=1,j
   ran_gn(j)=ran_gn(j)+b(j,i)*z(i)
 enddo
enddo
```

```
end function ran_gn
```

Here, we first call the subroutine `genera(b,c,n)` to generate the matrix `b` from the correlation matrix `c`, and then we can call the vector function `ran_gn(n,mu,b)` which returns the desired vector of Gaussian random numbers.

Note that the number of operations needed to carry out this process of generation of n Gaussian random variables increases as n^2. There are specific cases in which the efficiency of the generation can be greatly improved. This is discussed in Appendix B.

3.6 Rejection Methods

We have already seen in Section 3.3 a rejection method at work. The procedure we used was to propose a value of the random variable $\hat{\mathbf{x}}$ we wanted to sample, and then decide whether we keep this proposed value or not. We now study in depth some details of this kind of methods. We advance that rejection methods are the ones that are best suited to the generation of high (and not so high) dimensional sets of random variables. Still, for the sake of clarity, we analyze first an example with only one variable. Let us consider a random variable $\hat{\mathbf{x}}$ whose pdf is

$$f_{\hat{\mathbf{x}}}(x) = \begin{cases} C \exp\left(-\frac{x^2}{2}\right), & -1 \le x \le 1, \\ 0, & x \notin (-1, 1) \end{cases} \tag{3.84}$$

which is a cut-off Gaussian distribution limited to the interval $(-1, 1)$. It is possible to obtain the normalization constant from

$$1 = \int_{-\infty}^{\infty} dx f_{\hat{\mathbf{x}}}(x) = C \int_{-1}^{1} dx \, \exp\left(-\frac{x^2}{2}\right) = C \mathrm{erf}\left(1/\sqrt{2}\right) \sqrt{2\pi} \tag{3.85}$$

or $C = (\mathrm{erf}(1/\sqrt{2})\sqrt{2\pi})^{-1} = 0.584369\ldots$. However, we do not really need to know the value of the normalization constant C. As we will see, this is one of the nice features that make rejection methods so useful. Recall now what a pdf means: $f_{\hat{\mathbf{x}}}(x)$ is a measure of the probability of finding a value of the random variable around x. Hence, if, for example, $f_{\hat{\mathbf{x}}}(x_1) = 2f_{\hat{\mathbf{x}}}(x_2)$, then x_1 is twice as probable to appear as x_2. What we will do is to generate values uniformly in the interval $(-1, 1)$ and accept the proposed value x with a probability proportional to its pdf $f_{\hat{\mathbf{x}}}(x)$. It seems obvious that those values that make $f_{\hat{\mathbf{x}}}(x)$ larger will be accepted more often than those that make $f_{\hat{\mathbf{x}}}(x)$ small, which is, at least qualitatively, what we want to achieve. Summing up, we propose the following steps:

1) Propose a value x sampled from the $\hat{\mathbf{U}}(-1, 1)$, uniform distribution, that is, $x = 2u - 1$.
2) Accept that value with a probability $h(x)$, proportional to $f_{\hat{\mathbf{x}}}(x) \propto \exp(-\frac{x^2}{2})$.

In Step 2, it is important to stress that the acceptance probability $h(x)$ must be a number between 0 and 1. Recall that $f_{\hat{\mathbf{x}}}(x)$ is nonnegative and normalized, but otherwise it may take any value between 0 and ∞. The acceptance probability is $\alpha f_{\hat{\mathbf{x}}}(x)$, with α a real number carefully chosen to ensure that the resulting probability $h(x)$ indeed does not exceed the value 1. Otherwise, α is arbitrary. In the example we are considering, the resulting acceptance probability is $\alpha C \exp(-\frac{x^2}{2})$. As $\exp(-\frac{x^2}{2}) \le 1$, $\forall x$, all we need is that $\alpha C \le 1$ and we take the simplest (and, as

we will see, also the more convenient) choice $\alpha = 1/C$ so the acceptance probability of a proposed value x is $h(x) = \exp(-\frac{x^2}{2})$.

How is the acceptance process performed? This is a Bernoulli process: either we accept the proposed value with probability $\exp(-\frac{x^2}{2})$, or we do not. This decision is made by comparing a uniform $\hat{\mathbf{U}}(0,1)$ random number with the acceptance probability. If the random number is smaller than this probability, the proposed value is accepted. Otherwise, if the random number is larger than the acceptance probability, it is rejected. What do we do if we reject the value? Answer: propose a new one. This algorithm can be programmed as follows:

```
do
 x=2.0d0*ran_u()-1.0d0
 if(ran_u() < exp(-0.5d0*x*x)) exit
enddo
```

We see in this simple example the power of the rejection method. The algorithm is really simple and requires only two lines of code. Try to do the same using the method based on the inversion of the cumulative function or $x = F_{\hat{\mathbf{x}}}^{-1}(u)$. The only problem with the rejection method could be that the rejection step is taken too often, requiring a large number of trial proposals before a number is actually generated. We will learn how to compute this average acceptance probability, or the average number of proposals we must make before accepting one value.

Let us now define what we mean by a general rejection method. It is based on the two following steps:

1) Propose a value x for the random variable distributed according to a given probability density function $g(x)$.
2) Accept that value x with a probability $h(x)$. Recall that this probability must satisfy $0 \leq h(x) \leq 1, \ \forall x$.

To connect with the previous example, we had taken there

$$g(x) = \begin{cases} \frac{1}{2}, & x \in (-1,1) \\ 0, & x \notin (-1,1) \end{cases} \tag{3.86}$$

$$h(x) = \exp\left(-\frac{x^2}{2}\right). \tag{3.87}$$

For the acceptance step, we consider a Bernoulli random variable $\hat{\mathbf{B}}$ which takes the value $\hat{\mathbf{B}} = 1$ (acceptance) with probability $h(x)$ and the value $\hat{\mathbf{B}} = 0$ (rejection) with probability $1 - h(x)$. During the proposal and acceptance/rejection, we generate a two-dimensional random variable $(\hat{\mathbf{x}}, \hat{\mathbf{B}})$ with joint distribution $f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,n)$. The proposal $g(x)$ can give rise to two values of $\hat{\mathbf{B}}$, which implies $g(x) = f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x, \hat{\mathbf{B}} = 0) + f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x, \hat{\mathbf{B}} = 1)$. We are interested in those values of x that result after acceptance. They are distributed according to $f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{B}} = 1)$, which we want to be identical to the given $f_{\hat{\mathbf{x}}}(x)$ we wish to sample. According to the definition of conditional pdf, this distribution is given by

$$f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{B}}=1)=\frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{\text{Prob}(\hat{\mathbf{B}}=1)}=\frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{\int_{-\infty}^{\infty}f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)\,dx}. \tag{3.88}$$

According to our review for conditional pdfs, we have

$$h(x)=\text{prob}(\hat{\mathbf{B}}=1|x)=\frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,0)+f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}=\frac{f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)}{g(x)}. \tag{3.89}$$

Replacing $f_{\hat{\mathbf{x}},\hat{\mathbf{B}}}(x,1)=h(x)g(x)$ in (3.88), we obtain

$$f_{\hat{\mathbf{x}}}(x)=f_{\hat{\mathbf{x}}}(x|\hat{\mathbf{B}}=1)=\frac{h(x)g(x)}{\int_{-\infty}^{\infty}h(x)g(x)\,dx} \tag{3.90}$$

as the pdf resulting of the double proposal/acceptance steps. As it should be, this pdf is properly normalized.

Summing up, to sample $f_{\hat{\mathbf{x}}}(x)$ we need to find two functions $h(x)$ and $g(x)$ such that $f_{\hat{\mathbf{x}}}(x)$ can be split as $f_{\hat{\mathbf{x}}}(x)=Ag(x)h(x)$, where A is a constant. Note that this implies

$$g(x)h(x)f_{\hat{\mathbf{x}}}(y)=g(y)h(y)f_{\hat{\mathbf{x}}}(x),\qquad \forall x,y \tag{3.91}$$

which is the so-called detailed balance condition.[3)]

The efficiency of a rejection method depends on the average probability of acceptance ϵ. If ϵ is very small, then we need a large average number of trials to generate a single value of $\hat{\mathbf{x}}$. The average acceptance probability can be computed using

$$\epsilon=\int_{-\infty}^{\infty}P(\hat{\mathbf{B}}=1|x)g(x)\,dx=\int_{-\infty}^{\infty}h(x)g(x)\,dx=\langle h\rangle. \tag{3.92}$$

This implies that, given $g(x)$, the efficiency increases with increasing $h(x)$ and, hence, it is convenient to take $h(x)$ as large as possible, always keeping the condition $0\le h(x)\le 1$. The average acceptance probability is related to the normalization constant. In the previous example, we have

$$\epsilon=\int_{-1}^{1}\frac{1}{2}\exp\left(-\frac{x^2}{2}\right)dx=\frac{1}{2C}. \tag{3.93}$$

If we know $C=(\text{erf}(1/\sqrt{2})\sqrt{2\pi})^{-1}=0.584369\ldots$, we get $\epsilon=0.855624\ldots$, which is a large average acceptance probability. If we did not know C, we could derive it from a numerical estimation of ϵ.

Once we have understood how rejection methods work, we could have generated (3.84) using the splitting

$$g(x)=\frac{1}{\sqrt{2\pi}}\exp\left(-\frac{x^2}{2}\right),\quad x\in(-\infty,\infty), \tag{3.94}$$

$$h(x)=\begin{cases}1, & \text{if } x\in(-1,1),\\ 0, & \text{if } x\notin(-1,1).\end{cases} \tag{3.95}$$

3) The name is not by accident. As the generation of the different values of the random variable is done independently of each other, it can be thought, formally, as a homogeneous Markov chain in which the proposal pdf $f(x|y)=\dfrac{h(x)g(x)}{\int_{-\infty}^{\infty}h(x)g(x)\,dx}$ is independent of y. The stationarity condition (1.141) then leads trivially to the detailed balance condition.

If we multiply these two functions, we obtain $g(h)h(x) \propto f_{\hat{\mathbf{x}}}(x)$, which is all that we need. To implement this option, we follow the next steps: (i) generate x according to the usual (not truncated) Gaussian distribution $\hat{\mathbf{G}}(0,1)$ and (ii) accept it with probability 1 (i.e., accept it) if the proposed value for x belongs to the interval $[-1,1]$. The algorithm can be programmed as follows:

```
do
 x=ran_g()
 if(abs(x) < 1.0d0) exit
enddo
```

The average acceptance probability of this alternative method is

$$\epsilon = \int_{-1}^{1} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right) dx = \text{erf}\left(1/\sqrt{2}\right) = 0.682698\ldots, \tag{3.96}$$

which is less than with the previous algorithm. However, to see which method is more efficient, we would need to evaluate the average time needed to compute one random number and to perform the comparison step.

Let us see another example. We want to generate values of a random variable $\hat{\mathbf{x}}$ whose pdf is

$$f_{\hat{\mathbf{x}}}(x) = C \exp\left(-\frac{x^2}{2} - x^4\right), \qquad x \in (-\infty, \infty) \tag{3.97}$$

where C is the normalization constant given by $C = 2\sqrt{2}e^{-1/32}K_{1/4}(1/32) = 0.643162\ldots$. We need to split $f_{\hat{\mathbf{x}}}(x) \propto g(x)h(x)$, $g(x)$ being a pdf we know how to sample numerically and $0 \le h(x) \le 1$. The obvious choice is

$$g(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{x^2}{2}\right), \quad x \in (-\infty, \infty), \tag{3.98}$$

$$h(x) = \exp(-x^4). \tag{3.99}$$

So we propose a value x drawn from a $\hat{\mathbf{G}}(0,1)$ Gaussian distribution and accept it with probability $\exp(-x^4)$. The algorithm can be programmed as follows:

```
do
 x=ran_g()
 if (ran_u() < exp(-x**4) exit
enddo
```

The acceptance probability is

$$\epsilon = \int_{-\infty}^{\infty} dx\, g(x)h(x) = \frac{1}{C\sqrt{2\pi}} \approx 0.620283\ldots. \tag{3.100}$$

Let us see now an example with the two-dimensional distribution (3.56). We consider the following functions:

$$g(x_1, x_2) = 1/4 \ \text{ if } x_1 \in (-1,1), x_2 \in (-1,1), \tag{3.101}$$

$$h(x_1, x_2) = \begin{cases} 1, & \text{if } x_1^2 + x_2^2 \le 1, \\ 0, & \text{if } x_1^2 + x_2^2 > 1. \end{cases} \tag{3.102}$$

$g(x_1, x_2)$ is the pdf of a two-dimensional point (x_1, x_2) distributed uniformly in the square $x_1 \in (-1, 1), x_2 \in (-1, 1)$. This can be generated from two independent random variables u_1, u_2 uniformly distributed in the $(0, 1)$ interval: $x_1 = 2u_1 - 1, x_2 = 2u_2 - 1$. The proposed point is accepted (with probability 1) if it belongs to the circle $x_1^2 + x_2^2 \leq 1$, and rejected otherwise. The algorithm can be implemented as follows:

```
do
 x=2.0d0*ran_u()-1.0d0
 y=2.0d0*ran_u()-1.0d0
 r2=x*x+y*y
 if (r2 < 1.0d0) exit
enddo
r=sqrt(r2)
c=x/r
s=y/r
```

We have added the last three lines which allow the calculation of the sine and cosine of the angle θ in polar coordinates. We know that this angle is uniformly distributed in the $(0, 2\pi)$ interval. So these lines compute the sine and cosine of an angle uniformly distributed in the $(0, 2\pi)$ interval without ever calling the `sin` or `cos` functions. This trick is sometimes used to replace the lines

```
v=pi2*ran_u()
ran_gbmw=u*cos(v)
ran_gbmw=u*cos(v)
```

of the Box–Muller–Wiener algorithm by

```
ran_gbmw=u*x/r
ran_gbmw=u*y/r
```

Let us now consider an n-dimensional example. Imagine we need to compute the integral

$$I(n) = \frac{\int_{-\infty}^{\infty} dx_1 \cdots \int_{-\infty}^{\infty} dx_n e^{-(x_1^2+\cdots+x_n)^2/2-(x_1\cdots x_n)^4} \cos(x_1 \cdots x_n)}{\int_{-\infty}^{\infty} dx_1 \cdots \int_{-\infty}^{\infty} dx_n e^{-(x_1^2+\cdots+x_n)^2/2-(x_1\cdots x_n)^4}}. \tag{3.103}$$

Obviously, we interpret it as the average value $\langle G(x_1, \ldots, x_n) \rangle$ with respect to the pdf $f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_n)$, with

$$G(x_1, \ldots, x_n) = \cos(x_1 \cdots x_n), \tag{3.104}$$

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1, \ldots, x_n) = Ce^{-(x_1^2+\cdots+x_n)^2/2-(x_1\cdots x_n)^4} \tag{3.105}$$

where C is the normalization constant of the pdf. To generate values of the n-dimensional random variable $(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$, we split the pdf as

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n) \propto g(x_1,\ldots,x_n)h(x_1,\ldots,x_n) \tag{3.106}$$

$$g(x_1,\ldots,x_n) = \prod_{i=1}^{n}\left[\frac{1}{\sqrt{2\pi}}e^{-x_i^2/2}\right] \tag{3.107}$$

$$h(x_1,\ldots,x_n) = e^{-(x_1\cdots x_n)^4}. \tag{3.108}$$

Note that, indeed, $0 \leq h(x_1,\ldots,x_n) \leq 1$. The proposal $g(x_1,\ldots,x_n)$ corresponds to n independent Gaussian $\hat{\mathbf{G}}(0,1)$ distributions. The program to implement the rejection algorithm to compute this integral is as follows:

```
program rejection
implicit none
double precision r,s,b,p,g,ran_g,ran_u
double precision, parameter :: pi=3.141592653589793d0
integer, parameter :: n=2
integer M,i,j,ij,na
 M=10000000
 r=0.d0
 s=0.d0
 na=0
 do ij=1,M
   do
     b=1.0d0
     do i=1,n
       b=b*dran_g()
     enddo
     na=na+1
     if (dran_u() < exp(-b**4)) exit
   enddo
   g=cos(b)
   r=r+g
   s=s+g*g
 enddo
 p=dble(M)/na
 r=r/M
 s=sqrt((s/M-r*r)/M)
 write(6,*) n,r,s,p
end program rejection
```

Note that, besides the value of the integral and its error, we also write the average acceptance probability. This program runs without problems for any arbitrary value of n. Some results are as follows: $I(n=2) = 0.922467 \pm 0.000037$, $I(n = 10) = 0.993885 \pm 0.000011$, $I(n = 40) = 0.99999666 \pm 0.00000024$, which strongly suggest $\lim_{n\to\infty} I(n) = 1$, a result that can be confirmed analytically. The average acceptance probability increases with n as $\epsilon(n = 2) = 0.748$, $\epsilon(n = 10) = 0.977$, and $\epsilon(n = 40) = 0.999987$. This increase of ϵ with n, unfortunately, is not a typical feature of rejection methods. On the contrary, usually the average acceptance probability decreases with n and becomes very small for the values of n of interest.

For instance, consider the n-dimensional random variable $(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ uniformly distributed inside an n-dimensional sphere of radius 1 whose pdf is given in (3.68). If we use a simple rejection method where we propose a value for the ith coordinate x_i taken independently from a uniform distribution in $(-1, 1)$, and accept the proposed vector $(x_1, \ldots, x_n)$ only if it satisfies $x_1^2 + \cdots + x_n^2 \leq 1$, then the average acceptance probability is $\epsilon(n) = V_n/2^n$, as 2^n is the volume of the n-dimensional hypercube $[-1, 1]^n$. This takes values $\epsilon(n = 2) = 0.7854\ldots$, $\epsilon(n = 10) = 2.49 \times 10^{-3}$, and $\epsilon(n = 100) = 1.87 \times 10^{-70}$. We see that, for $n = 100$, we need to make, on average, 10^{70} proposals to accept one! This is clearly not useful. The method of rejection needs to be modified in order to be able to deal with distributions like this one.

Let us see another example for which the performance of rejection methods greatly decreases with the number of variables. We consider the so-called ϕ^4 distribution of interest in the field of statistical mechanics of phase transitions, which we will discuss in detail in Chapter 5. In the particular version we consider here, we have n random variables $(\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_n)$ whose joint pdf is

$$f_{\hat{\mathbf{x}}_0,\ldots,\hat{\mathbf{x}}_n}(x_1, \ldots, x_n) = C \exp[-V(x_1, \ldots, x_n)], \tag{3.109}$$

with[4)]

$$V(x_1, \ldots, x_n) = \sum_{i=1}^{n} \left[\frac{1}{2} \left((x_{i+1} - x_i)^2 + a x_i^2 \right) + b x_i^4 \right]. \tag{3.110}$$

To sample this distribution, one might be tempted to split the pdf as

$$f_{\hat{\mathbf{x}}_0,\ldots,\hat{\mathbf{x}}_n}(x_1, \ldots, x_n) = C \exp[-\mathcal{L}_0(x_1, \ldots, x_n)] \times \exp\left[-b \sum_{i=1}^{n} x_i^4 \right] \tag{3.111}$$

and then generate a set of correlated Gaussian variables with pdf

$$g(x_1, \ldots x_n) = C \exp[-\mathcal{L}_0(x_1, \ldots, x_n)] \tag{3.112}$$

using the quadratic form (called the "free Lagrangian" in some contexts)

$$\mathcal{L}_0(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i=1}^{n} \left[(x_{i+1} - x_i)^2 + a x_i^2 \right] \tag{3.113}$$

using the method explained in Section 3.5 (or the more sophisticated one developed in Appendix B), and then accept that proposal with probability

$$h(x_1, \ldots, x_n) = \exp\left[-b \sum_{i=1}^{n} x_i^4 \right]. \tag{3.114}$$

The problem with this procedure is that the average acceptance probability greatly decreases (in fact, exponentially) with n (see Exercise 18). For moderate values of n (say, $n = 100$), this acceptance probability is so small that not a single value is effectively accepted in the lifetime of the person running the simulation.

4) In the sum, we use the "periodic boundary conditions" convention, by which x_{n+1} is equivalent to x_1.

As in previous cases, it is possible to generalize the rejection methods to discrete distributions. If we propose values of a random variable distributed according to a discrete pdf

$$g(x) = \sum_i g_i \delta(x - x_i) \tag{3.115}$$

and then accept the proposed value x_i with probability h_i (fulfilling, of course, the condition $0 \leq h_i \leq 1,\ \forall i$), then the resulting distribution of this proposal/acceptance method is

$$f_{\hat{\mathbf{x}}}(x) = \frac{\sum_i h_i g_i \delta(x - x_i)}{\sum_i h_i g_i}. \tag{3.116}$$

So, if we want to sample a discrete distribution $f_{\hat{\mathbf{x}}}(x) = \sum_i p_i \delta(x - x_i)$, we can split $p_i \propto h_i g_i$ with the conditions $0 \leq h_i, g_i \leq 1$ and $\sum_i g_i = 1$. We then propose a value x_i according to the distribution $g(x)$ and accept it with probability h_i.

Let us see an example. We want to generate values of a discrete random variable $\hat{\mathbf{x}}$ which takes integer values with probability

$$\text{Prob}(\hat{\mathbf{x}} = i) = p_i = \frac{-\log(1-a)}{a} \frac{a^i}{1+i}, \qquad i = 0, 1, 2, \ldots \tag{3.117}$$

with $0 < a < 1$ being a given number. The obvious split in this case is $g_i = (1-a)a^i$, a geometric distribution $\hat{\mathbf{N}}_G(p)$ of the parameter $p = 1 - a$, and $h_i = (1+i)^{-1}$, which fulfills $0 \leq h_i \leq 1$. The algorithm would be as follows:

```
do
 iran_f=dlog(ran_u())/dlog(a)
 if (ran_u() < 1.0d0/(1.0d0+iran_f)) exit
enddo
```

Unfortunately, not many discrete distributions g_i can be generated easily and serve as proposal for the rejection step. A simple idea, though, allows us to use a continuous random variable as a proposal suitable for the generation of a discrete distribution. The point is to relate the discrete variable $\hat{\mathbf{x}}$ we want to sample to a suitable continuous variable $\hat{\mathbf{x}}_c$. For the sake of concreteness, we assume that $\hat{\mathbf{x}}$ takes only nonnegative integer values $0, 1, 2, \ldots$, and let $p_i = \text{Prob}(\hat{\mathbf{x}} = i)$. We define a continuous random variable $\hat{\mathbf{x}}_c$ via the pdf defined as $f_{\hat{\mathbf{x}}_c}(x) = p_{[x]}$, being, as usual, $[x]$, the integer part of x. For instance, if $p_i = \frac{6}{\pi^2}(1+i)^{-2}$ for $i = 0, 1, 2, \ldots$, we define $f_{\hat{\mathbf{x}}_c}(x) = \frac{6}{\pi^2}(1+[x])^{-2}$, $x \geq 0$. It should be clear now that $f_{\hat{\mathbf{x}}_c}(x)$ is properly normalized and that the corresponding cumulative function $F_{\hat{\mathbf{x}}_c}(x)$ satisfies $F_{\hat{\mathbf{x}}_c}(i+1) - F_{\hat{\mathbf{x}}_c}(i) = p_i$. The relation between the two random variables is simply $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_c]$, as can be seen by checking that $\text{Prob}(\hat{\mathbf{x}} = i) = \text{Prob}(i \leq \hat{\mathbf{x}}_c < i+1) = F_{\hat{\mathbf{x}}_c}(i+1) - F_{\hat{\mathbf{x}}_c}(i) = p_i$, see Figure 3.2.

Therefore, in order to obtain values of the discrete random variable $\hat{\mathbf{x}}$, all we need to do is to generate values of the continuous random variable $\hat{\mathbf{x}}_c$ and set $\hat{\mathbf{x}} = [\hat{\mathbf{x}}_c]$.

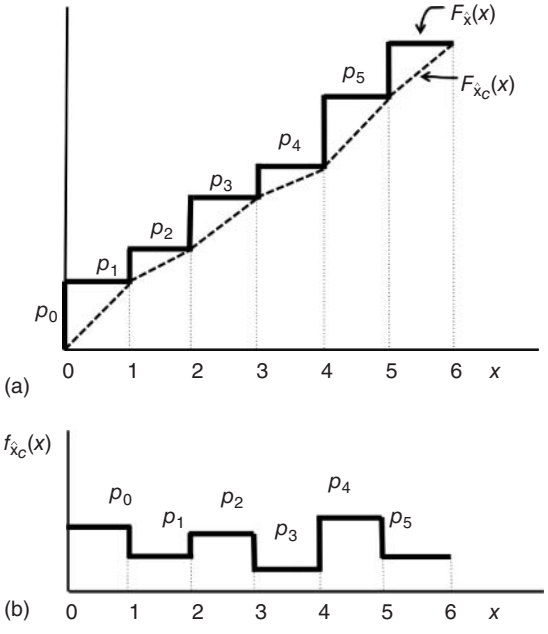


Figure 3.2 (a) Piecewise linear approximation $F_{\hat{\mathbf{x}}_c}(x)$ (dotted line) to the true cdf (solid line) of the discrete variable $F_{\hat{\mathbf{x}}_c}(x)$. (b) Corresponding $f_{\hat{\mathbf{x}}_c}(x)$.

If we use a rejection method for the generation of $\hat{\mathbf{x}}_c$, the process proceeds as for any standard continuous distribution: propose a value x from a pdf $g(x)$ and accept it with probability $h(x) = Cf_{\hat{\mathbf{x}}_c}(x)/g(x)$, choosing C such that $0 \le h(x) \le 1$. If accepted, the integer value $[x]$ of the obtained variable will give us values of the discrete random variable $\hat{\mathbf{x}}$.

Let us give an example. Consider again the integer distribution $p_i = \frac{6}{\pi^2}(1+i)^{-2}$ for $i = 0, 1, 2, \ldots$. This was done before in Section 2.4, using a direct method. We introduce the random variable $\hat{\mathbf{x}}_c$ whose pdf is $f_{\hat{\mathbf{x}}_c}(x) = \frac{6}{\pi^2}(1+[x])^{-2}, x \ge 0$. For the proposal distribution, we choose a pdf $g(x)$ as close as possible to $f_{\hat{\mathbf{x}}_c}(x)$ but such that it can be easily generated. A natural choice (but not the only possible one) is $g(x) = (1+x)^{-2}, x \ge 0$. The cdf is $G(x) = \dfrac{x}{1+x}$, and the solution of $G(x) = u$ is $x = \frac{u}{1-u}$. This proposed value is accepted with a probability $h(x) = Cf_{\hat{\mathbf{x}}_c}(x)/g(x)$. It turns out that $C = 1/4$ keeps the limits $0 \le h(x) \le 1$, so we choose

$$h(x) = \frac{1}{4}\left(\frac{1+x}{1+[x]}\right)^2.$$

If x is accepted, then its integer part $i = [x]$ is distributed according to p_i. The function can be implemented as follows:

```
integer function iran_pot2()
implicit none
double precision :: ran_u,u,x,h,acc
```

```
do
 u=ran_u()
 x=u/(1.0d0-u)
 h=0.25d0*((1.0d0+x)/(1+int(x)))**2
 if (ran_u() < h) exit
enddo
iran_pot2=int(x)
```

```
end function iran_pot2
```

Let us apply this method to the discrete Poisson distribution $p_i = e^{-\lambda}\frac{\lambda^i}{i!}$. We begin by defining a continuous random variable $\hat{\mathbf{x}}_c$ whose pdf is

$$f_{\hat{\mathbf{x}}_c}(x) = e^{-\lambda}\frac{\lambda^{[x]}}{[x]!}. \tag{3.118}$$

Now we need a good proposal $g(x)$. We know that $g(x)$ has to be as close as possible to $f_{\hat{\mathbf{x}}_c}(x)$. We focus now on the case of large λ for which previous algorithms were not so efficient. In this case, $f_{\hat{\mathbf{x}}_c}(x)$ comes close to a Gaussian distribution of mean and variance equal to λ. It would seem that a good choice would be the Gaussian distribution

$$g(x) = \frac{1}{\sqrt{2\pi\lambda}}e^{-\frac{(x-\lambda)^2}{2\lambda}}. \tag{3.119}$$

However, it turns out that the acceptance probability $h(x) = Cf_{\hat{\mathbf{x}}_c}(x)/g(x)$ cannot satisfy the condition $h(x) \leq 1$, as the ratio $f_{\hat{\mathbf{x}}_c}(x)/g(x)$ tends to infinity with x. Instead, and based on the shape of $f_{\hat{\mathbf{x}}_c}(x)$, we propose the Cauchy distribution

$$g(x) = \frac{1}{\pi\sqrt{\lambda}}\frac{1}{1+\left(\frac{x-\lambda}{\sqrt{\lambda}}\right)^2} \tag{3.120}$$

which has a maximum at $x = \lambda$ and a width of the order of $\lambda^{1/2}$. The acceptance probability is

$$h(x) = Ce^{-\lambda}\frac{\lambda^{[x]}}{[x]!}\pi\sqrt{\lambda}\left(1+\left(\frac{x-\lambda}{\sqrt{\lambda}}\right)^2\right), \qquad x \geq 0. \tag{3.121}$$

The constant C is chosen under the condition $h(x) \leq 1, \forall x$. Based on the Stirling approximation at $x = \lambda$, $\lambda! \approx e^{-\lambda}\lambda^\lambda\sqrt{2\pi\lambda}$, we adopt $C = 1/\sqrt{\pi}$, which, as a simple graphical analysis shows, indeed fulfills the condition $h(x) \leq 1$ if $\lambda \geq 20$. In the cases of $\lambda < 20$, we can use other methods for the Poisson distribution. To generate a value x from the above written Cauchy distribution, we set $x = \lambda + \lambda^{1/2}z$ with $z = \tan(\pi u)$. It is convenient to write the acceptance probability as

$$h(x) = (1+z^2)\sqrt{\pi\lambda}\exp\left[-\lambda + [x]\log\lambda - \log([x]!)\right] \tag{3.122}$$

since good routines exist for the calculation of the logarithm of the factorial.[5] The acceptance probability of this algorithm is $C \approx 0.564$. A possible implementation is as follows:

5) For example, one could use the function `gammln(x+1)` from [5].

```
integer function iran_poisson(lambda)
implicit none
double precision :: ran_u,lambda,h,z
  do
    z=dtan(3.141592653589793d0*ran_u())
    iran_poisson=lambda+sqrt(lambda)*z
    if (iran_poisson >= 0) then
      h=(1.0d0+z*z)*sqrt(pi*lambda)*&
      dexp(-lambda+iran_poisson*dlog(lambda)-&
      gammln(dble(iran_poisson+1)))
      if (ran_u() < h) exit
    endif
  enddo
end function iran_poisson
```

Further Reading and References

An explicit algorithm to generate Gaussian random numbers using a numerical inversion method is provided in [6].

A rejection method (known as the "ziggurat"), which corrects the systematic errors coming from the piecewise linear approximation used in the numerical inversion and is suitable for generating values of a monotonically decreasing pdf, has been developed in [7].

A distribution similar to (3.34) was actually needed in a problem related to field theories for systems with absorbing states [8].

Exercises

1) Check that the transformation $x = -\log(u)$ generates random numbers with pdf $f_{\hat{\mathbf{x}}}(x) = \exp(-x),\ x \geq 0$, if u is uniformly distributed in the $(0, 1)$ interval. Do this by generating M random numbers and approximating the pdf $f_{\hat{\mathbf{x}}}(x)$ by a histogram of width Δx. Use $M = 10^4$ and $\Delta x = 0.01$ and check the dependence on M and Δx.
2) Repeat the previous exercise for the Gaussian $\hat{\mathbf{G}}(0, 1)$ distribution using (i) the Box–Muller–Wiener algorithm, (ii) the approximate algorithm inverse error function, and (iii) the linear interpolation approximation. Check in each case the quality of the obtained distribution and the computer time needed.
3) Use the numerical inversion method to generate random numbers according to the pdf $f_{\hat{\mathbf{x}}}(x) = \frac{3}{4}(1 - x^2),\ x \in (-1, 1)$. Compare the pdf with the histogram of the obtained numbers.
4) For the pdf $f_{\hat{\mathbf{x}}}(x) = nx^{n-1},\ x \in (0, 1)$, compare the efficiency of the algorithms $x = \max(u_1, \ldots, u_n)$ and $x = u^{1/n}$ as a function of n.
5) Use a rejection method to sample the pdf $f_{\hat{\mathbf{x}}}(x) = Ce^{-\frac{1}{2}x^2},\ x \in (0, 1)$, and use the result to evaluate the integral of Exercise 2.10 with the sampling method.

6) Let $x_1, \ldots, x_6$ be Gaussian random numbers of zero mean and correlations $C_{ij} = \frac{1}{2}\delta_{i-1,j} + \delta_{i,j} + \frac{1}{2}\delta_{i+1,j}$ (assume periodic boundary conditions, that is, replace 7 by 1 and 0 by 6 whenever these numbers appear). Generate six-dimensional number using this distribution and use those numbers to compute the average value $\langle x_1 x_2 x_3 x_4 x_5 x_6 \rangle$, and compare with the exact result given by Wick's theorem.
7) Plot pairs of numbers (x, y) obtained as $x = r\cos(\theta)$, $y = r\sin(\theta)$, with r and θ being random variables uniformly distributed in the $(0, 1)$ and $(0, 2\pi)$ intervals, respectively, and check graphically that these numbers are not uniformly distributed in the unit circle of center $(0, 0)$ and radius 1. Design a rejection algorithm that produces points uniformly distributed in the unit circle.
8) Design a rejection algorithm that yields random numbers uniformly distributed in the surface of an n-dimensional sphere of radius 1. Compute the rejection probability of the algorithm as a function of n.
9) Design a rejection method to generate values of a random variable $\hat{\mathbf{x}}$ distributed according to the pdf

$$f_{\hat{\mathbf{x}}}(x) = \frac{x^{\alpha-1}\mathrm{e}^{-x}}{\Gamma(\alpha)}, \qquad , x \geq 0$$

valid for $0 < \alpha < 1$.
10) Use a rejection method to generate values of the random variable $\hat{\mathbf{x}}$ distributed according the pdf $f_{\hat{\mathbf{x}}}(x) = C\exp(-\frac{1}{2}x^2 - x^4)$. Compute numerically, from the acceptance probability, the normalization constant C.
11) Repeat the previous problem for the pdf $f_{\hat{\mathbf{x}}}(x) = C\exp(\frac{1}{2}x^2 - x^4)$.
12) Use the numerical inversion of the cumulative distribution function to generate values of a discrete Poisson random variable of parameter λ. How long does it take to generate 10^6 numbers for λ=0.5, 1, 2, 5, and 10?
13) Repeat the previous problem using a rejection method using a convenient geometric distribution as proposal.
14) Generate the binomial distribution using (i) the numerical inversion of the distribution function and (ii) a rejection method.
15) Implement a numerical inversion method to generate values of the two-dimensional random variable $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ distributed according to

$$f_{\hat{\mathbf{x}}\hat{\mathbf{y}}}(x, y) = \begin{cases} 6x, & \text{if } x + y \leq 1, x \geq 0, y \geq 0, \\ 0, & \text{otherwise.} \end{cases}$$

16) Repeat the previous exercise using a rejection method.
17) Use a rejection technique with the proposal choice

$$g(x) = \frac{1}{\sqrt{2a\lambda}} e^{-\sqrt{\frac{2}{a\lambda}}|x-\lambda|}$$

and a suitable value of a to generate Poisson-distributed random numbers. Compare its efficiency with the Cauchy proposal given in the main text.

18) Measure the acceptance probability of the rejection method proposed to sample the multidimensional pdf (3.109) and (3.110) and study its dependence on the number of variables n. Get a rough estimate of that dependence by replacing $\sum_{i=1}^{n} x_i^4$ by its average value $\sum_{i=1}^{n} \langle x_i^4 \rangle = n\langle x^4 \rangle = 3n\langle x^2 \rangle^2$ assuming a Gaussian approximation.

4
Dynamical Methods

Simple rejection methods for high-dimensional probability distributions have, in general, a low acceptance probability—so low, in fact, that it renders them unfeasible to sample most distributions of interest. In this chapter, we develop a variant of the rejection method that allows us, finally, to sample almost any probability distribution using simple rules. The method is that of rejection *with repetition*. Of course, there is no free lunch, and there is an important price to pay when using these methods: as they produce correlated (as opposed to independent) values of the random variables, it turns out that the error of the estimate of the integral or sum increases with N, the number of variables. We will see, though, that the increase of the error with N is moderate as compared to the decrease of the average acceptance probability if a simple rejection method is used.

4.1
Rejection with Repetition: a Simple Case

Our goal here is, again, to devise methods to generate values x_k that can be used in a sampling method in order to compute a given integral

$$I = \int dx f_{\hat{\mathbf{x}}}(x) G(x) \tag{4.1}$$

using the sample mean and variance, (2.11) and (2.12). For that purpose, we need to generate values x_k of a random variable distributed according to the given pdf $f_{\hat{\mathbf{x}}}(x)$. We propose now a modification of the rejection method, which we name *rejection with repetition*. The modification is such that every time a proposal is rejected, instead of proposing a new value, we simply return the value that had been generated in the previous step. We will explain exactly what we mean using an example. Imagine we want to sample values from the one-variable pdf

$$f_{\hat{\mathbf{x}}}(x) = C \exp\left[-\frac{1}{2}x^2 - x^4\right]. \tag{4.2}$$

We know already how to do this using rejection methods: propose a value of x according to a Gaussian $\hat{\mathbf{G}}(0,1)$ distribution, and accept it with probability $h(x) = e^{-x^4}$. We even wrote a short program to do this. An equivalent version of the program, more suitable for our purposes here, is:

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

```
1 x=ran_g()
 if (ran_u() > exp(-x**4)) goto 1
```

The idea of the repetition is to avoid going back whenever a value is rejected at the kth trial; instead, the previous value of the random variable, the one obtained at the $(k-1)$th trial, is kept. This can be programmed as follows:

```
xp=ran_g()
if (ran_u() < exp(-xp**4)) x=xp
```

Here, `xp` is the proposed value that we accept with probability `exp(-x**4)`. If the value is not accepted, we do not try to propose a new value, but the current value of x (the one obtained in the previous call) is not modified. We just need to take into account that, just in case the trial at the first initial step fails, the value of x should have been initialized following some distribution $f_{\hat{\mathbf{x}}_0}(x)$ (e.g., a Gaussian distribution), so we need to add, before the first attempt to generate a random number and only then, a line like

```
x=ran_f0()
```

where `ran_f0()` returns a random variable distributed according to some distribution $f_{\hat{\mathbf{x}}_0}(x)$, or simply

```
x=x0
```

with `x0` some initial value. This corresponds to taking $f_{\hat{\mathbf{x}}_0}(x) = \delta(x - x_0)$.

How does the output of such a routine looks like? Since we are repeating values when rejecting the proposed value, a typical output is

```
k         x_k
0 -0.38348165934483291
1 -0.38348165934483291
2 -0.38348165934483291
3 -0.38348165934483291
4 0.47884603175682383
5 0.47884603175682383
6 0.37469679082063412
7 0.35122027008152767
8 -0.44596168309186857
9 -0.44596168309186857
10 -0.44596168309186857
```

We do not need much knowledge of statistics to see that in this list some numbers are not independent; in fact they are equal! The question is whether we can still use this list of numbers in the estimation of the integral (4.1). Put in another way, is it true that the series of numbers x_k are still distributed according to $f_{\hat{\mathbf{x}}}(x)$ as given by (3.90)? As one value x_{k+1} might be equal to the previous value x_k, we need to find the probability distribution that governs the outcome of the kth step of this process.

Let $\hat{\mathbf{x}}_k$ be the random variable obtained during the kth step. Its precise form will depend on the outcome of the acceptance process. As there are two options,

namely accept or reject, we consider the acceptance step to be a Bernoulli variable $\hat{\mathbf{y}}$ which can take two possible values $\hat{\mathbf{y}} = 1$, accept, and $\hat{\mathbf{y}} = 0$, reject. Direct application of (1.135) leads to

$$f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}_k}(x|\text{accept})\, p(\text{accept}) + f_{\hat{\mathbf{x}}_k}(x|\text{reject})\, p(\text{reject}), \tag{4.3}$$

with $p(\text{accept})$ and $p(\text{reject}) = 1 - p(\text{accept})$ being the probabilities of acceptance and rejection, respectively, at step k. Now we can use the equivalent of (1.128) in the case of a discrete random variable $\hat{\mathbf{y}}$ to modify the first term of this sum:

$$f_{\hat{\mathbf{x}}_k}(x) = \text{Prob}(\text{accept}|x)\, g(x) + f_{\hat{\mathbf{x}}_k}(x|\text{reject})\, (1 - p(\text{accept})), \tag{4.4}$$

where we have used that $g(x)$ is the pdf of proposing the value x at step k. The probability of accepting a given value x is $\text{Prob}(\text{accept}|x) = h(x)$. If we reject, the pdf at step k is the same as the valid one at step $k-1$, that is, $f_{\hat{\mathbf{x}}_k}(x|\text{reject}) = f_{\hat{\mathbf{x}}_{k-1}}(x)$. Finally, the acceptance probability is given by (3.92). This leads to

$$f_{\hat{\mathbf{x}}_k}(x) = h(x)g(x) + f_{\hat{\mathbf{x}}_{k-1}}(x) \left[1 - \int_{-\infty}^{\infty} h(x)g(x)\, dx\right]. \tag{4.5}$$

This recurrence relation gives $f_{\hat{\mathbf{x}}_k}(x)$ in terms of $f_{\hat{\mathbf{x}}_{k-1}}(x)$. We should not be confused by this simple linear recurrence relation. The solution, in terms of the initial distribution $f_{\hat{\mathbf{x}}_0}(x)$, is

$$f_{\hat{\mathbf{x}}_k}(x) = (1-\epsilon)^k \left[f_{\hat{\mathbf{x}}_0} - \frac{h(x)g(x)}{\int_{-\infty}^{\infty} h(x)g(x)\, dx}\right] + \frac{h(x)g(x)}{\int_{-\infty}^{\infty} h(x)g(x)\, dx}, \tag{4.6}$$

where we have written $\epsilon = \int_{-\infty}^{\infty} h(x)g(x)dx$, the average acceptance probability. This relation can be written in terms of the pdf $f_{\hat{\mathbf{x}}}(x)$ we want to sample from as given by 3.90:

$$f_{\hat{\mathbf{x}}_k}(x) = (1-\epsilon)^k \left[f_{\hat{\mathbf{x}}_0} - f_{\hat{\mathbf{x}}}(x)\right] + f_{\hat{\mathbf{x}}}(x). \tag{4.7}$$

Then the answer to the question: are all the x_k values distributed according to $f_{\hat{\mathbf{x}}}(x)$? is, generally, NO, as we cannot conclude from this formula that $f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}}(x), \forall k$. However, if the initial numbers are already distributed according to $f_{\hat{\mathbf{x}}}(x)$, that is, if $f_{\hat{\mathbf{x}}_0}(x) = f_{\hat{\mathbf{x}}}(x)$, then we find $f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}}(x), \forall k$ and the numbers we obtain by this procedure are indeed distributed according to the desired distribution. If, on the other hand, the initial pdf $f_{\hat{\mathbf{x}}_0}(x)$ is not equal to the one we want to sample, the situation is not so bad either. According to (4.7), the difference between the actual distribution at step k, $f_{\hat{\mathbf{x}}_k}(x)$, and the one we want to sample, $f_{\hat{\mathbf{x}}}(x)$, monotonically decreases with k. In fact, as $0 < \epsilon \leq 1$, we have the exact result, independent of the initial distribution $f_{\hat{\mathbf{x}}_0}$:

$$\lim_{k\to\infty} f_{\hat{\mathbf{x}}_k}(x) = f_{\hat{\mathbf{x}}}(x). \tag{4.8}$$

Imagine $\epsilon = 0.5$. After $k = 10$ steps, the factor $(1-\epsilon)^k \approx 10^{-3}$. If $k = 100$, $(1-\epsilon)^k \approx 10^{-30}$, an absolutely negligible contribution in most cases. Even for a low acceptance probability $\epsilon = 10^{-2}$, it takes about $k = 2300$ steps to have $(1-\epsilon)^k \approx 10^{-10}$. Therefore, in order to ensure that the produced x_k values satisfy the desired distribution, we need to discard some steps at the beginning. This is

the process called *thermalization*. How many steps M_0 we should discard depends on the distance between the initial and the desired distributions and the average acceptance probability ϵ.

The reader will have noticed that the process of generation of the numbers x_k can be understood in terms of a Markov chain, since the result at step k depends only on the distribution at step $k-1$. It is easy to write the recurrence relation 4.5 in the form of the recurrence equation (1.140) for a homogeneous Markov chain,

$$f_{\tilde{\mathbf{x}}_n}(x) = \int_{-\infty}^{\infty} f(x|y) f_{\tilde{\mathbf{x}}_{n-1}}(y)\, dy, \tag{4.9}$$

with a transition probability density function

$$f(x|y) = h(x)g(x) + \left[1 - \int_{-\infty}^{\infty} h(x)g(x)\, dx\right] \delta(x-y). \tag{4.10}$$

The two terms of the sum on the right-hand side correspond to the two possibilities, namely acceptance of a proposed value or its rejection.

4.2 Statistical Errors

Remember the approximation consisting of replacing an average value by the sample mean

$$I = \int_{-\infty}^{\infty} G(x) f_{\tilde{\mathbf{x}}}(x)\, dx = \hat{\boldsymbol{\mu}}_M[G] \pm \sigma[\hat{\boldsymbol{\mu}}_M] \tag{4.11}$$

with the sample average

$$\hat{\boldsymbol{\mu}}_M[G] = \frac{1}{M} \sum_{k=1}^{M} G(x_k). \tag{4.12}$$

As usual, as dictated by Chebyshev's theorem, the error of the sample mean is measured by its standard deviation. However, we cannot use now the relation

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{\sigma^2[G]}{M} \approx \frac{\hat{\boldsymbol{\sigma}}_M^2[G]}{M} \tag{4.13}$$

because the derivation of this formula assumes that all contributions to the sum in (4.12) are independent of each other and we know this is not the case for the rejection with repetition method. Let us get now a useful expression for the error in the case of correlated values. We assume that we have thermalized conveniently the algorithm by discarding the necessary steps at the beginning and the Markov chain is in the steady state and, consequently, all variables x_k are distributed according to the same pdf $f_{\tilde{\mathbf{x}}}(x)$.

The variance of the sample mean is defined as

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \langle (\hat{\boldsymbol{\mu}}_M - I)^2 \rangle. \tag{4.14}$$

Replacing $\hat{\boldsymbol{\mu}}_M$ from (4.12) and writing G_k instead of $G(x_k)$ for brevity of notation, we obtain

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \left\langle \left[\frac{1}{M} \sum_{k=1}^{M} (G_k - I) \right]^2 \right\rangle$$
$$= \frac{1}{M^2} \sum_{k=1}^{M} \sum_{j=1}^{M} \Big\langle (G_k - I)(G_j - I) \Big\rangle . \tag{4.15}$$

In this double sum, we consider separately the terms with (i) $k = j$, (ii) $k > j$, and (iii) $j > k$. Owing to the symmetry between k and j, the last two contributions are equal and we can write

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{1}{M^2} \sum_{k=1}^{N} \langle (G_k - I)^2 \rangle + \frac{2}{M^2} \sum_{k=1}^{M-1} \sum_{j=k+1}^{M} \langle (G_k - I)(G_j - I) \rangle. \tag{4.16}$$

Expanding the product in the second sum and using that $\langle G_i \rangle = I, \ \forall i$, we get

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{1}{M^2} \sum_{k=1}^{M} \left[\langle G_k^2 \rangle - I^2 \right] + \frac{2}{M^2} \sum_{k=1}^{M-1} \sum_{j=k+1}^{M} \left[\langle G_k G_j \rangle - I^2 \right]$$
$$= \frac{\sigma^2[G]}{M} \left[1 + \frac{2}{M} \sum_{k=1}^{M-1} \sum_{j=k+1}^{M} \rho_G(k,j) \right] \tag{4.17}$$

where we have used that all G_k yield the same variance:

$$\sigma^2[G] = \langle G_k^2 \rangle - I^2 \tag{4.18}$$

and the definition of the normalized correlation between G_k and G_j

$$\rho_G(k,j) = \frac{\langle G_k G_j \rangle - I^2}{\sigma^2[G]}. \tag{4.19}$$

As we are in the steady state of a homogeneous Markov chain, the correlation function depends only on the difference $j - k \equiv i$, $\rho_G(k,j) = \rho_G(j-k) = \rho_G(i)$, and we can write

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{\sigma^2[G]}{M} \left[1 + \frac{2}{M} \sum_{k=1}^{M-1} \sum_{i=1}^{M-k} \rho_G(i) \right] . \tag{4.20}$$

Exchanging the summation order, we get

$$\sum_{k=1}^{M-1} \sum_{i=1}^{M-k} \rho_G(i) = \sum_{i=1}^{M-1} \sum_{k=1}^{M-i} \rho_G(i) = \sum_{i=1}^{M-1} (M-i) \rho_G(i) \tag{4.21}$$

since the sum over k could be performed as $\rho_G(i)$ does not depend on k. We have now the final result

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{\sigma^2[G]}{M} \left[2 \sum_{i=1}^{M-1} \left(1 - \frac{i}{M} \right) \rho_G(i) + 1 \right] . \tag{4.22}$$

Let us now define the autocorrelation time τ_G as

$$\tau_G = \sum_{i=1}^{M-1} \left(1 - \frac{i}{M} \right) \rho_G(i). \tag{4.23}$$

We obtain the final expression for the variance of the sample mean, again replacing the true variance $\sigma^2[G]$ by the sample variance $\hat{\sigma}^2_M[G]$:

$$\sigma^2[\hat{\boldsymbol{\mu}}_M] = \frac{\hat{\sigma}^2_M[G]}{M}(2\tau_G + 1). \tag{4.24}$$

Let us stress that the only assumption in this definition is that the Markov chain is in the steady state where variables at each step k have the same distribution and the correlations between steps k and j depend only on the difference $j - k$ (see exercise 2). For instance, if all variables $\hat{\mathbf{x}}_k$ were independent, we would have $\rho_G(i) = \delta_{i,0}$, the autocorrelation time would be 0, and the variance would simplify to the known result $\sigma^2[\hat{\boldsymbol{\mu}}_M] = \sigma^2[G]/M$. In general, the autocorrelation function $\rho_G(i)$ decays with i and tends to 0 as $i \to \infty$ (usually, but not always, the decay is exponential).

The autocorrelation time τ_G is a measure of the decay of the correlation function. It can be defined, loosely, as the number of steps that are needed for the correlation to decay significantly from the initial value $\rho_G(0) = 1$. A more precise definition is simply (4.23). For instance, if the decay were truly exponential $\rho_G(i) = [\rho_G(1)]^i$, with $\rho_G(1) < 1$, then according to the definition (4.23) the correlation time is

$$\tau_G = \sum_{i=1}^{M-1} \left(1 - \frac{i}{M}\right) [\rho_G(1)]^i = \frac{\rho_G(1)}{1 - \rho_G(1)} - \frac{\rho_G(1)(1 - [\rho_G(1)]^M)}{(1 - \rho_G(1))^2 M} \tag{4.25}$$

which can be reduced to

$$\tau_G = \frac{\rho_G(1)}{1 - \rho_G(1)} \tag{4.26}$$

in the limit of large M. Indeed, in most cases, the correlation function decays rapidly and, in the limit of large M, we can neglect the factor i/M in the definition of τ_G and write

$$\tau_G = \sum_{i=1}^{\infty} \rho_G(i). \tag{4.27}$$

For the rejection with repetition method explained above, it is possible to compute the correlation function. If the acceptance probability is ϵ, then either $G_{i+k} = G_k$ with probability $(1 - \epsilon)^i$ (all i steps in going from k to $i + k$ have been rejections) or G_{i+k} is independent of G_k with probability $1 - (1 - \epsilon)^i$. This gives

$$\langle G_k G_{i+k} \rangle = (1 - \epsilon)^i \langle G_k^2 \rangle + [1 - (1 - \epsilon)^i] \langle G_k \rangle \langle G_{i+k} \rangle \tag{4.28}$$

which, replaced in the definition (4.19), leads to $\rho_G(i) = (1 - \epsilon)^i$, which is an exponential decay. The autocorrelation time is independent of the function G and is given by $\tau_G = \dfrac{1 - \epsilon}{\epsilon}$, and then $2\tau_G + 1 = \dfrac{2 - \epsilon}{\epsilon}$ and the correct expression for the estimator and its error is

$$I = \hat{\boldsymbol{\mu}}_M[G] \pm \frac{\hat{\boldsymbol{\sigma}}_M[G]}{\sqrt{M}} \sqrt{\frac{2 - \epsilon}{\epsilon}}. \tag{4.29}$$

Note that this is equivalent to using an effective number $M_{\text{eff}} = M \dfrac{\epsilon}{2 - \epsilon} \leq M$ of independent contributions to the estimator. If we wanted to have the same error

using a rejection method without repetition (i.e., trying again if the proposed value is rejected), then we would need an average number M_{eff}/ϵ of proposals. As $\frac{2-\epsilon}{\epsilon} \geq \frac{1}{\epsilon}$, it turns out that the method of rejection with repetition needs to generate more random numbers in order to yield the same statistical error. However, we will see in Section 4.3 that this method is the one that can be generalized to high-dimensional distributions while keeping a reasonable acceptance probability.

4.3 Dynamical Methods

Rejection methods, irrespective of whether we use repetition or not, can have a very low acceptance probability. This is especially true if the proposal probability $g(x)$ is very different from $f_{\hat{\mathbf{x}}}(x)$ and does not reflect the strong variations in value that $f_{\hat{\mathbf{x}}}(x)$ might have, that is, the ratio $f_{\hat{\mathbf{x}}}(x_{\max})/f_{\hat{\mathbf{x}}}(x_{\min})$. This usually occurs when we have no clue on how the pdf $f_{\hat{\mathbf{x}}}(x)$ we want to sample looks like, a problem particularly severe and common for distributions taking values in a high N-dimensional space, $x = (x_1, \ldots, x_N)$.

Let us give a specific example. Consider the distribution

$$f_{\hat{\mathbf{x}}}(x_1, \ldots, x_N) = \begin{cases} C \exp\left(- \sum_{i=1, j=i}^{N} x_i x_j \right), & \text{if } x_i \in (-1, 1), \ \forall i, \\ 0, & \text{if } x_i \notin (-1, 1). \end{cases} \tag{4.30}$$

One might be tempted to use a rejection method in which the proposal are independent values $x \equiv (x_1, \ldots, x_N)$, each x_i drawn from a $\hat{\mathbf{U}}(-1, 1)$ or uniformly distributed in the interval $(-1, 1)$. This proposal would be accepted with a probability $h(x_1, \ldots, x_N)$ proportional to $f_{\hat{\mathbf{x}}}(x_1, \ldots, x_N)$. We note that the maximum value of the distribution $f(x_1, \ldots, x_N)$ occurs for $x_i = 0, \forall i$ and there are two equivalent minima for $x_i = 1, \forall i$, and $x_i = -1, \forall i$[1]. As there are a total of $N(N+1)/2$ terms in the sums of the exponential defining $f_{\hat{\mathbf{x}}}(x)$, the ratio between these two extreme values is $f_{\hat{\mathbf{x}}}(x_{\max})/f_{\hat{\mathbf{x}}}(x_{\min}) = \mathrm{e}^{N(N+1)/2}$. The optimal acceptance probability would be $h(x_1, \ldots, x_N) = \exp(-\sum_{i=1, j=i}^{N} x_i x_j)$, whose minimum and maximum values are $e^{-N(N+1)/2}$ and 1, respectively. When N is large, it turns out that most of the proposed values will belong to a region in which the acceptance probability is very small. Hence, the average acceptance probability is very small. This can be checked with the following program:

```
program average
implicit none
integer, parameter :: n=100
integer m,ijk,i,j
double precision :: x(n)
double precision :: averh,sum,ran_u
```

1) It might help to derive these results from the identity $\sum_{i=1, j=i}^{N} x_i x_j = \frac{1}{2}\left(\left(\sum_{i=1}^{N} x_i \right)^2 + \sum_{i=1}^{N} x_i^2 \right)$.

```
m=1000000
averh=0.0d0
do ijk=1,m
 do i=1,n
   x(i)=2.d0*ran_u()-1.0d0
 enddo
 sum=0.0d0
 do i=1,n
  do j=i,n
    sum=sum+x(i)*x(j)
  enddo
 enddo
  averh=averh+dexp(-sum)
enddo
write(6,*) n,averh/m
end program average
```

On running this program with different values of N, we obtain that the average acceptance probability ϵ decreases with N. In fact, it is found that the average acceptance probability decays exponentially with N (see Problem 1). For example, for $N = 10$, it is $\epsilon \approx 0.105$; for $N = 20$, $\epsilon = 1.68 \times 10^{-2}$; and for $N = 100$, $\epsilon \approx 3 \times 10^{-8}$, which means that we have to propose, on average, around 30 million numbers, in order to accept one, which is a very inefficient procedure indeed.

How can we avoid proposing values that have a very low acceptance probability? The idea of the so-called dynamical methods is to use the information gathered at previous proposals to make a new proposal. In this way, when by repeated trials we reach the region in which the pdf is high, our proposals will not tend to abandon this region. Let us explain these ideas using again a simpler example than the N-dimensional distribution (4.30). Consider the cutoff Gaussian probability distribution

$$f_{\tilde{\mathbf{x}}}(x) = C \exp\left(-x^2/2\sigma^2\right), \quad x \in (-1, 1) \tag{4.31}$$

for small values of the parameter σ[2]. If we propose "blindly" values uniformly distributed according to a uniform distribution $g(x)$ in the interval $(-1, 1)$ and accept this value with a probability $h(x) = \exp\left(-x^2/2\sigma^2\right)$, most of the times we are proposing values that have a very small acceptance probability[3]. However, when we reach a value near $x = 0$ and note that there is a high acceptance probability, what we should do is that the new proposed value is again close to 0, as we know now (and did not know before) that $x = 0$ is the region of high probability. If x_k is the value obtained at the kth step, then the new proposal at step $k + 1$ could be, for example, a number x_{k+1} chosen uniformly in the interval $x_{k+1} \in (x_k - \delta, x_k + \delta)$, with δ a number comparable to σ, so we do not leave the region of large probability. This is the basic principle behind dynamical methods.

Summing up, in the static methods explained in Chapter 3, the new value for the random variable was chosen each time independently of previous values, whereas

2) σ is not exactly equal to the standard deviation, as the distribution does not extend for all real values of x.

3) The average acceptance probability is $\epsilon = \sqrt{\frac{\pi}{2}}\sigma \mathrm{erf}(1/\sigma\sqrt{2})$. This tends to $\epsilon \to \sqrt{\frac{\pi}{2}}\sigma$ as $\sigma \to 0$.

in the dynamical methods the proposal (and the acceptance probability as we will see) depends on the previous values of the random variable. In other words, in the static methods, if we denote by x_n the value of the random variable generated at the nth step, then both the proposal $g(x)$ and the acceptance $h(x)$ are independent of the previous values $(x_1, \dots, x_{n-1})$. As a consequence, the pdf $f_{\hat{\mathbf{x}}_n}(x)$ at step n is independent of the previous steps. More precisely, in the static methods

$$f_{\hat{\mathbf{x}}_n}(x_n|x_0, x_1, \dots, x_{n-1}) = f_{\hat{\mathbf{x}}_n}(x_n) = f_{\hat{\mathbf{x}}}(x_n), \quad n = 0, 1, 2, \dots . \tag{4.32}$$

and all random variables x_n are independently distributed according to the same pdf $f_{\hat{\mathbf{x}}}(x)$.

In a dynamical method, both the proposal and the acceptance probabilities depend on the previous results. As a consequence, the pdf of the random variable x_n obtained in the nth proposal/acceptance step depends on the previous values of the random variables $x_0, x_1, \dots, x_{n-1}$. The simplest way to implement this dependence is via the latest result, that is, take

$$f_{\hat{\mathbf{x}}_n}(x_n|x_0, \dots, x_{n-1}) = f_{\hat{\mathbf{x}}_n}(x_n|x_{n-1}) \tag{4.33}$$

defining our series of proposal/acceptance steps as a Markov chain. Again, the simplest implementation uses a homogeneous Markov chain where the conditional probabilities are independent of the step, or $f_{\hat{\mathbf{x}}_n}(x_n|x_{n-1}) = f(x_n|x_{n-1})$.

Let us now go into the details of a dynamical method. We perform a realization of the Markov chain generating values $x_0, x_1, \dots, x_n, \dots$ of the random variables. The value x_n is sampled from a distribution $f(x_n|x_{n-1})$. Ideally, we would like all probability distributions $f_{\hat{\mathbf{x}}_n}(x)$ to be equal to the desired pdf $f_{\hat{\mathbf{x}}}(x)$. However, our discussion of the simple dynamical method of rejection with repetition tells us that it might be more reasonable to demand only that the asymptotic distribution reproduces the desired pdf, $\lim_{n\to\infty} f_{\hat{\mathbf{x}}_n}(x) = f_{\hat{\mathbf{x}}}(x)$. Our discussion on Markov chains in Section 1.9 tells us that a sufficient condition for this to happen is the detailed balance condition

$$f(y|x) f_{\hat{\mathbf{x}}}(x) = f(x|y) f_{\hat{\mathbf{x}}}(y). \tag{4.34}$$

Usually (but not always), the generation of random variables distributed according to $f(x|y)$ uses a rejection method. This means that we propose a value x from a pdf $g(x|y)$, and then accept it with a probability $h(x|y)$. As explained, both proposal and acceptance of a number x at step n depend on the value y obtained in the previous step. We now obtain the transition probability $f(x|y)$ that corresponds to this proposal/acceptance algorithm in the case of rejection with repetition, that is, if the value x is not accepted, then we keep the previous value y. It will be clear in a moment why, of all possible rejection methods, we adopt this one. The derivation of the transition pdf is similar to the one leading to (4.10), but now we have to take into account that both the proposal pdf $g(x|y)$ and the acceptance probabilities $h(x|y)$ depend on y.

The transition pdf $f(x|y)$ has two contributions corresponding to the acceptance or not of the proposed value. Given a value y, the overall rejection probability is

$$P(\text{rejection}|y) = 1 - P(\text{acceptance}|y)$$
$$= 1 - \int h(z|y)g(z|y)\,dz \equiv 1 - \epsilon(y) \tag{4.35}$$

where we have used (3.92) with the only modification that both the proposal and the acceptance probabilities now depend on the value y. If the proposal x is not accepted, then the transition probability is the Dirac-delta $\delta(x-y)$, as we keep the old value y. This leads to a transition probability generalizing (4.10):

$$f(x|y) = h(x|y)g(x|y) + \delta(x-y)\left[1 - \epsilon(y)\right]. \tag{4.36}$$

Similarly, we have

$$f(y|x) = h(y|x)g(y|x) + \delta(x-y)\left[1 - \epsilon(x)\right]. \tag{4.37}$$

Which functions $h(x|y)$ and $g(x|y)$ can be used? The only requirement is that the stationary pdf of this Markov chain is the distribution $f_{\hat{\mathbf{x}}}(x)$. We enforce this by imposing the sufficient condition of detailed balance[4]. Replacing $f(x|y)$ and $f(y|x)$ in the detailed balance condition (4.34), the two terms with delta-functions are identical as they force $x = y$, leading to

$$h(y|x)g(y|x)f_{\hat{\mathbf{x}}}(x) = h(x|y)g(x|y)f_{\hat{\mathbf{x}}}(y) \tag{4.38}$$

as the equation to be satisfied by the proposal $g(x|y)$ and the acceptance $h(x|y)$ to ensure that the stationary distribution of the Markov chain is indeed $f_{\hat{\mathbf{x}}}(x)$.

It is essential to return the value $x_n = x_{n-1}$ if the proposal x is not accepted instead of keeping on proposing new values until they get accepted as we used to do in a standard rejection method. The reason is simple to understand. If we sampled $f(x|y)$ without repetition, that is, trying new values until one is accepted, the same argument that led to the distribution (3.90) tells us that the resulting pdf is

$$f(x|y) = \frac{h(x|y)g(x|y)}{\int dz\, h(z|y)g(z|y)}. \tag{4.39}$$

And, similarly

$$f(y|x) = \frac{h(y|x)g(y|x)}{\int dz\, h(z|x)g(z|x)}. \tag{4.40}$$

The detailed balance condition to be satisfied by the proposal $g(x|y)$ and the acceptance $h(x|y)$ probabilities is

$$\frac{f_{\hat{\mathbf{x}}}(y)h(x|y)g(x|y)}{\int dz\, h(z|y)g(z|y)} = \frac{f_{\hat{\mathbf{x}}}(x)h(y|x)g(y|x)}{\int dz\, h(z|x)g(z|x)}. \tag{4.41}$$

And, because of the integrals, it is difficult to give general solutions for $g(x|y)$ and $h(x|y)$ satisfying this equation.

4) We are not aware of any general solution that does not use the detailed balance condition.

We write now in full the recurrence relation of the Markov chain:

$$f_{\hat{\mathbf{x}}_{n+1}}(x) = \int f(x|y) f_{\hat{\mathbf{x}}_n}(y)\, dy \tag{4.42}$$
$$= \int h(x|y) g(x|y) f_{\hat{\mathbf{x}}_n}(y)\, dy + f_{\hat{\mathbf{x}}_n}(x) \left[1 - \int dz h(z|x) g(z|x)\right]$$

where

$$\epsilon(x) = \int dz\, h(z|x) g(z|x) \tag{4.43}$$

is the acceptance probability given x. The overall average acceptance probability is

$$\epsilon = \langle \epsilon(x) \rangle = \int dx\, \epsilon(x) f_{\hat{\mathbf{x}}}(x). \tag{4.44}$$

We will review now different solutions to the detailed balance condition in its form (4.38). There are, mainly, two ways to find the solution of this equation: the Metropolis *et al.* algorithm, and the heat-bath algorithm.

4.4 Metropolis *et al.* Algorithm

In the Metropolis *et al.* algorithm, we fix the proposal distribution $g(x|y)$ and solve for the acceptance probability. The proposal $g(x|y)$ is arbitrary insofar the resulting algorithm is ergodic. In particular, it is required that all possible values for the random variable $\hat{\mathbf{x}}$ have the chance to be reached eventually by the Markov chain. Given $g(x|y)$ we must find then a function $h(x|y)$ satisfying

$$0 \leq h(x|y) \leq 1 \tag{4.45}$$
$$\frac{h(x|y)}{h(y|x)} = q(x|y) \tag{4.46}$$

where we have introduced

$$q(x|y) = \frac{g(y|x) f_{\hat{\mathbf{x}}}(x)}{g(x|y) f_{\hat{\mathbf{x}}}(y)} \tag{4.47}$$

which satisfies the condition

$$q(y|x) = \frac{1}{q(x|y)}. \tag{4.48}$$

To find all solutions of (4.46) we introduce the function $\omega(x, y)$ by means of

$$h(x|y) = \sqrt{q(x|y)}\, \omega(x, y) \tag{4.49}$$

which, when replaced in (4.46), requires the symmetry condition $\omega(x, y) = \omega(y, x)$. Condition (4.45) is more delicate. A possibility is to demand that $\omega(x, y)$ depends on x and y through the function $q(x|y)$: $\omega(x, y) = \omega(q(x|y))$, with the symmetry condition $\omega(x, y) = \omega(y, x)$ being equivalent to $\omega(q) = \omega(1/q)$. The requirement (4.45) then leads to $0 \leq \sqrt{q}\omega(q) \leq 1$ or $\omega(q) \leq 1/\sqrt{q}$, from where $\omega(q) = \omega(1/q) \leq 1/\sqrt{1/q}$. We are looking, then, for functions $\omega(q)$ satisfying

$$\omega(q) = \omega(1/q), \tag{4.50}$$

$$\omega(q) \leq \sqrt{q}. \tag{4.51}$$

The first solution is the one given by Metropolis *et al.*:

$$\omega(q) = \min\left(\sqrt{q}, 1/\sqrt{q}\right). \tag{4.52}$$

It satisfies trivially $\omega(q) = \omega(1/q)$. For condition (4.51), we take separately the cases $q \leq 1$ and $q \geq 1$. If $q \leq 1$, then $\omega(q) = \sqrt{q}$. For $q > 1$, it is $\omega(q) = 1/\sqrt{q} \leq \sqrt{q}$. This choice (4.52) leads to one of the most widely used solutions of the detailed balance condition:

$$h(x|y) = \min\left(1, q(x|y)\right). \tag{4.53}$$

The second most widely used solution is that of Glauber:

$$\omega(q) = \frac{1}{\sqrt{q} + \sqrt{1/q}} \tag{4.54}$$

which trivially satisfies (4.50) and (4.51). In terms of the transition probability, we have

$$h(x|y) = \frac{q(x|y)}{1 + q(x|y)}. \tag{4.55}$$

Another solution was used later by van Beijeren and Schulman. It is simply $\omega(q) = C$, a constant. The constant has to be chosen such that $C \leq \max_{x,y} \sqrt{q(y|x)}$, in order to fulfill the condition $h(x|y) \in [0, 1]$.

Let us now apply the Metropolis *et al.* algorithm to some specific examples.

4.4.1 Gaussian Distribution

We consider the following random variable $\hat{\mathbf{x}}$ whose pdf is Gaussian:

$$f_{\hat{\mathbf{x}}}(x) = C \exp\left(-\frac{x^2}{2}\right). \tag{4.56}$$

The normalization constant C is irrelevant for the Metropolis algorithm. We take the following proposal probability:

$$g(x|y) = \begin{cases} \frac{1}{2\Delta}, & |x - y| < \Delta, \\ 0, & \text{otherwise.} \end{cases} \tag{4.57}$$

In other words, x is drawn from a uniform distribution in the interval $(y - \Delta, y + \Delta)$. The constant Δ is arbitrary for now, but later we will determine which is the best choice for it. Obviously, the proposal satisfies the symmetry condition

$$g(x|y) = g(y|x) \tag{4.58}$$

so the function $q(x|y)$ is

$$q(x|y) = \frac{g(y|x) f_{\hat{\mathbf{x}}}(x)}{g(x|y) f_{\hat{\mathbf{x}}}(y)} = \exp\left(\frac{y^2 - x^2}{2}\right). \tag{4.59}$$

We now need an acceptance probability $h(x|y)$. We will use Metropolis *et al.*'s choice $h(x|y) = \min(1, q(x|y))$. In our example, it is

$$q(x|y) = \begin{cases} 1, & |y| > |x|, \\ \exp\left(\frac{y^2-x^2}{2}\right), & |y| \leq |x|. \end{cases} \tag{4.60}$$

As usual, the acceptance step is a Bernoulli process with probability q. We draw a $\hat{\mathbf{U}}(0,1)$ random number u and compare it with q. If $u \leq q$, we accept the proposal. Note that if $q = 1$, the proposal is always accepted and there is no need to spend time drawing a random number and comparing it to 1. The algorithm can be summarized as follows:

1) Given y propose x uniformly from $(y - \Delta, y + \Delta)$.
2) If $|x| \leq |y|$, accept x.
3) If $|x| > |y|$, accept x with probability $q = \exp(\frac{y^2-x^2}{2})$. If the value x is rejected, return y as the value of the Gaussian variable.

The program could look like this:

```
double precision function ran_gauss_mc(y,delta)
implicit none
double precision:: x,y,delta,ran_u
x=y+delta*(2.0d0*ran_u()-1.0d0)
if(abs(x) > abs(y)) then
 if (ran_u() > exp(0.5d0*(y-x)*(y+x))) then
! Reject. Do not accept the proposed value. Keep old one.
  ran_gauss_mc=y
  return
 endif
endif
y=x
ran_gauss_mc=y
end function ran_gauss_mc
```

Before the first call to this routine, we need to set an initial value for y, for instance, $y = 0$.

We can understand intuitively the way the algorithm works. If the proposal tends to increase the probability, that is, if $f_{\hat{\mathbf{x}}}(x) \geq f_{\hat{\mathbf{x}}}(y)$, then the new value is accepted. This moves the Markov chain toward values of high probability. Still, if $f_{\hat{\mathbf{x}}}(x) < f_{\hat{\mathbf{x}}}(y)$, the new value is not systematically rejected, so there is a chance of visiting regions of low probability. As we have shown, the resulting distribution of values of x coming from these steps (unconditional acceptance of values with larger probability, and conditional acceptance otherwise) is precisely $f_{\hat{\mathbf{x}}}(x)$.

There is still the issue of the parameter Δ. Notice, first, that whatever the value of Δ, the proposal $g(x|y)$ given by (4.57) leads to an ergodic algorithm because it allows each and every real value x to be proposed eventually and it cannot be trapped in loops. It is a random walk in the real space which does not have any forbidden region. Certainly, some values will be accepted more often than others, but the proposal probability is such that all values of x could be visited at one time

or another. Note that the average acceptance probability ϵ depends on Δ. Using (4.44), and after a lengthy calculation, we obtain

$$\epsilon = 1 - \mathrm{erf}\left(\frac{\Delta}{2\sqrt{2}}\right) + \frac{4}{\Delta\sqrt{2\pi}}\left(1 - e^{-\Delta^2/8}\right) \tag{4.61}$$

which monotonically decreases to 0 as Δ increases from its maximum value $\epsilon = 1$ at $\Delta = 0$. On the basis of this result, one might think, wrongly, that the smaller the Δ the more efficient the algorithm, as the acceptance probability is larger. However, it is clear that for a very small value of Δ, the proposed value of x will be very close to the old value y, and hence there will be a large correlation between x and y, the old and the proposed (and very likely to be accepted) value, respectively. This induces a large correlation time and an increase in the errors. Instead, we come to the general idea that, in order to optimize the algorithm, the parameter Δ has to be chosen such that the correlation time τ_G (and hence the statistical errors) reaches its minimum value. We will come to this important point later.

Here, we can check what would happen if we did not repeat the previous value in case of rejection and insisted on proposing a new value until it got accepted, as we used to do in a standard rejection method.

```
double precision function ran_gauss_mc_bad(y,delta)
implicit none
double precision:: x,y,delta,ran_u
1 x=y+delta*(2*ran_u()-1)
if(abs(x) > abs(y)) then
 if (ran_u() > exp(0.5d0*(y-x)*(y+x))) goto 1
endif
y=x
ran_gauss_mc_bad=y
end function ran_gauss_mc_bad
```

The reader can check that this routine does not produce numbers distributed according to a Gaussian distribution (it becomes worse as Δ increases).

4.4.2 Poisson Distribution

We give an example now of the application of the Metropolis *et al.* algorithm to a discrete distribution. We consider a Poisson random variable that takes integer values $\hat{\mathbf{x}} = k,\ k = 0, 1, 2, 3, \ldots$ with probability

$$p_k = e^{-\lambda}\frac{\lambda^k}{k!}. \tag{4.62}$$

We shall proceed very similarly to the Gaussian distribution. Given a current value k, we must propose a new value k' sampled from a distribution $g(k'|k)$. We propose to use

$$g(k'|k) = \begin{cases} 1/2, & k' = k+1, \\ 1/2, & k' = k-1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.63}$$

We only propose values k' that differ from k by one unit. In this way, we obtain an ergodic algorithm: all integer values have, in principle, the chance to be proposed at one time or another. Note, furthermore, that this proposal satisfies the symmetry condition $g(k'|k) = g(k|k')$.

With the choice (4.63), the function $q(k'|k)$ is

$$q(k'|k) = \frac{p_{k'}}{p_k} = \lambda^{k'-k}\frac{k!}{k'!}. \tag{4.64}$$

And the acceptance probability $h(k'|k) = \min(1, q(k'|k))$ can take two possible values:

a) If $k' = k+1$, then

$$q(k+1|k) = \frac{\lambda}{k+1} \to h(k+1|k) = \begin{cases} 1, & \lambda \geq k+1, \\ \frac{\lambda}{k+1}, & \lambda < k+1. \end{cases} \tag{4.65}$$

b) If $k' = k-1$, then

$$q(k-1|k) = \frac{k}{\lambda} \to h(k-1|k) = \begin{cases} \frac{k}{\lambda}, & \lambda \geq k, \\ 1, & \lambda < k. \end{cases} \tag{4.66}$$

(Note, in particular, that $h(-1|0) = 0$. So, the value -1 can be proposed but will never be accepted). The Metropolis *et al.* algorithm works in the following way:

- Chose randomly with probability $1/2$, $k' = k+1$ or $k' = k-1$.
- If $k' = k+1$, then if $k' \leq \lambda$ accept k'; if $k' > \lambda$ accept k' with probability $\frac{\lambda}{k'}$.
- If $k' = k-1$, then if $k \geq \lambda$ accept k'; if $k < \lambda$ accept k' with probability $\frac{k}{\lambda}$.

Here is a possible version of the program:

```
integer function iran_poisson_mc(lambda)
  implicit none
  double precision lambda,dran_u
  integer, save :: k=0
  integer :: k1
  if (dran_u().gt.0.5d0) then
    k1=k+1
    if (k1.le.lambda) then
      k=k1
    elseif (dran_u().lt.lambda/k1) then
      k=k1
    endif
  else
    k1=k-1
    if (k.ge.lambda) then
      k=k1
    elseif (dran_u().lt.k/lambda) then
      k=k1
    endif
  endif
  iran_poisson_mc=k
end function iran_poisson_mc
```

The average acceptance probability could be computed in principle, using the discrete version of (4.44) with sums replacing integrals, but it is simpler to compute

it numerically using a simple modification of the previous program. A numerical analysis concludes that $\epsilon \approx 1 - 0.4\lambda^{-1/2}$ is a good fit to the numerical data. Here, we do not have any parameters than can be tuned, but these could be introduced, for instance, by setting a proposal distribution $g(k'|k) = 1/(2L+1)$ if $|k' - k| \leq L$. The integer number L could then be used to reduce the correlation time. We leave this as an exercise to the reader.

4.5 Multidimensional Distributions

The two previous applications have served as an introduction to the Metropolis *et al.* algorithm. However, its real power lies in the sampling of high-dimensional distributions where the random variable has N components, with a large N. We will use the notation $x = (s_1, s_2, \ldots, s_N)$ to denote the N-dimensional random variable $\hat{\mathbf{x}}$ we need to sample. Let us begin by an explicit pdf for $N = 4$:

$$\begin{aligned} f_{\hat{\mathbf{x}}}(s_1, s_2, s_3, s_4) &= A \exp\left(-s_1^4 - s_2^4 - s_3^4 - s_4^4 + s_1 s_2 + s_2 s_3 + s_3 s_4 + s_4 s_1\right) \\ &\equiv A e^{d(x)} \end{aligned} \tag{4.67}$$

where A is an irrelevant normalization factor and we have defined $d(x) = -s_1^4 - s_2^4 - s_3^4 - s_4^4 + s_1 s_2 + s_2 s_3 + s_3 s_4 + s_4 s_1$. The dynamical method proposes a new value $x' = (s_1', s_2', s_3', s_4')$ drawn from a distribution $g(x'|x)$ and accepts it with probability $h(x'|x)$. A simple extension of the previous examples suggests the proposal

$$\begin{aligned} s_1' &= s_1 + (2u_1 - 1)\Delta, \\ s_2' &= s_2 + (2u_2 - 1)\Delta, \\ s_3' &= s_3 + (2u_3 - 1)\Delta, \\ s_4' &= s_4 + (2u_4 - 1)\Delta \end{aligned} \tag{4.68}$$

with u_1, u_2, u_3, and u_4 being independent $\hat{\mathbf{U}}(0,1)$ variables. This means that each proposal s_i' is drawn from a uniform distribution in $(s_i - \Delta, s_i + \Delta)$. Note that this proposal is symmetrical as $g(x'|x) = g(x|x')$. The acceptance probability can be chosen as $h(x|x') = \min(1, q(x|y)$, with

$$q(x'|x) = \frac{f_{\hat{\mathbf{x}}}(x')}{f_{\hat{\mathbf{x}}}(x)} = e^{d(x') - d(x)}. \tag{4.69}$$

Hence, $h(x'|x) = 1$ if $d(x') > d(x)$, and $h(x'|x) = e^{d(x') - d(x)}$, otherwise. A routine to implement the generation of this four-dimensional variable $\hat{\mathbf{x}}$ could be as follows:

```
subroutine ran_f4(s1,s2,s3,s4,delta)
implicit none
double precision:: s1,s2,s3,s4,q1,q2,q3,q4,d1,d2,delta,ran_u
q1=s1+delta*(2*ran_u()-1)
q2=s2+delta*(2*ran_u()-1)
q3=s3+delta*(2*ran_u()-1)
q4=s4+delta*(2*ran_u()-1)
d1=-s1**4-s2**4-s3**4-s4**4+s1*s2+s2*s3+s3*s4+s4*s1
```

```
d2=-q1**4-q2**4-q3**4-q4**4+q1*q2+q2*q3+q3*q4+q4*q1
if(d2 < d1) then
 if (ran_u() > exp(d2-d1)) return
endif
s1=q1
s2=q2
s3=q3
s4=q4
return
end
```

A `call ran_f4(s1,s2,s3,s4,delta)` returns in (s_1, s_2, s_3, s_4) a value of the four-dimensional random variable $\hat{\mathbf{x}}$. Obviously, it is possible to replace

```
if(d2 < d1) then
 if (ran_u() > exp(d2-d1)) return
endif
```

by the single line

```
if (ran_u() > exp(d2-d1)) return
```

as the condition inside the `if` will never be satisfied for $d_1 < d_2$. However, this replacement implies computing every time an exponential and generating a random number whether we need it or not, and the three-line code is more efficient, albeit not very clear. The average acceptance probability depends on Δ. It would be difficult to compute it analytically, but the numerical results tell us that the average acceptance decays exponentially with the parameter Δ. This simply means, again, that the farther we allow the proposal to be with respect to the actual value of the variable, the smaller the acceptance ratio and, hence, the more often the routine will return exactly the same value for the random number.

It should be clear now the great advantage of this algorithm: it can be easily generalized to the case in which the dimensionality N of the random variable $\hat{\mathbf{x}}$ or the actual number of variables $(s_1, s_2, \ldots, s_N)$ is large. And by large we mean any number that can be accommodated in the computer memory. Let us be more precise and consider the pdf

$$f_{\hat{\mathbf{x}}}(s_1, \ldots, s_N) = A \exp\left(\sum_{i=1}^{N} \left[-s_i^4 + s_i s_{i+1} \right] \right). \tag{4.70}$$

We have used the convention, named "periodic boundary conditions," which, whenever they appear, s_{N+1} must be replaced by s_1, and (this will be needed later) s_0 must be replaced by s_N, as if the variables were lying in a ring. It should be clear now the generalization of the Metropolis *et al.* algorithm, as shown in the following program:

```
subroutine ran_fn(s,n,d,delta)
implicit none
integer n,i,i1
double precision, dimension (n):: s,q
double precision:: d,dn,delta,ran_u
```

```
dimension s(n),q(n)
do i=1,n
 q(i)=s(i)+delta*(2*ran_u()-1)
enddo
dn=0.0d0
do i=1,n
 i1=i+1
 if (i.eq.n) i1=1
 dn=dn-q(i)**4+q(i)*q(i1)
enddo
if(dn < d) then
 if (ran_u() > exp(dn-d)) return
endif
s=q
d=dn
return
end
```

Notice that we keep the value of the function $d(x)$, so we only need to compute its new value. The value of d must be initialized before the first call to this subroutine.

If we fix now the value of Δ, say $\Delta = 0.1$, the average acceptance decays with N. For instance, for $N = 10$, it is $\epsilon \approx 0.852$, whereas for $N = 10^3$ it is $\epsilon \approx 4.8 \times 10^{-2}$ and for $N = 10^4$ it is $\epsilon \approx \times 10^{-6}$. It is clear then, that the average probability can become very small for large N. We could compensate this by decreasing Δ with N such that the average acceptance probability remains approximately constant.

It is true that a low acceptance probability is not, per se, a fundamental problem. What we care about is the correlation time τ_G, as the error increases with τ_G according to (4.24). But we believe, and we will sustain this belief later, that a small acceptance probability implies a large correlation time because many a time the values of the random variable will be identical as the new proposals get rejected once and again.

The standard way of having a constant average acceptance with N consists in modifying the proposal $g(x'|x)$ in such a way that only one of the N variables $(s_1, \dots, s_N)$ is proposed to change. Formally, we use a proposal pdf

$$g(x'|x) = \sum_{i=1}^{N} \frac{1}{N} g_i(s_i'|s_i) \prod_{j \neq i} \delta(s_j' - s_j). \tag{4.71}$$

This means that we select randomly one of the s_i variables, $i = 1, \dots, N$ (each one with probability $1/N$), propose a value s_i' from a probability distribution $g_i(s_i'|s_i)$, and keep all other variables with $j \neq i$ unchanged. For the single-variable proposal probability, we take the same as before:

$$g_i(s_i'|s_i) = \frac{1}{2\Delta}, \quad \text{if } |s_i' - s_i| < \Delta \tag{4.72}$$

or s_i' chosen randomly and uniformly from the interval $(s_i - \Delta, s_i + \Delta)$. It is clear that this proposal satisfies the symmetry condition $g(x'|x) = g(x|x')$, and hence the function $q(x'|x)$ is given by the ratio $f_{\hat{\mathbf{x}}}(x')/f_{\hat{\mathbf{x}}}(x)$. What is more interesting is that the calculation of this ratio implies only a few variables since most of them are unchanged and simplify in the numerator and denominator. Specifically, we have

$$q(x'|x) = \frac{f_{\hat{\mathbf{x}}}(s_1, \ldots, s'_i, \ldots, s_N)}{f_{\hat{\mathbf{x}}}(s_1, \ldots, s_i, \ldots, s_N)} = \exp\left[-s_i'^4 + s_i^4 + (s_{i-1} + s_{i+1})(s'_i - s_i)\right] \tag{4.73}$$

(recall our periodic boundary conditions convention: $s_0 \equiv s_N$ and $s_{N+1} \equiv s_1$).

Using the choice $h(x'|x) = \min(1, q(x'|x))$ for the acceptance probability, the algorithm would be as follows:

```
subroutine ran_fn_1(s,n,delta)
implicit none
integer k,n,i,ip,im,i_ran
double precision, dimension (n) :: s
double precision::q,d,delta,ran_u
do k=1,n
 i=i_ran(n)
 q=s(i)+delta*(2*ran_u()-1)
 ip=i+1
 if (i.eq.n) ip=1
 im=i-1
 if (i.eq.1) im=n
 d=-q**4+s(i)**4+(s(im)+s(ip))*(q-s(i))
 if(d < 0) then
   if (ran_u() > exp(d)) cycle
 endif
 s(i)=q
enddo
return
end
```

Here we have used the function `i_ran(n)` which returns an integer number uniformly distributed between `1` and `n`. This is easily implemented as the integer part of `n*ran_u()+1`, being `ran_u()`, as usual, a uniform random number in the interval `(0,1)` (and avoiding the exact value 1, see the note 4 in page 45). When we take, for example, $\Delta = 2.0$, the average acceptance probability is $\epsilon \approx 0.43$ independently on the value of N, however large this would be.

Note that in the previous program we have repeated the basic proposal/acceptance step N times using the loop `do k=1,n`. This makes sense, as in every single proposal/acceptance step we only modify at most one of the N variables and do not change the other $N-1$ variables. This repetition of N times the basic step is what is called *one Monte Carlo step* or 1 MCS. A new value of the N-dimensional random variable $\hat{\mathbf{x}}$ is only returned after each Monte Carlo step.

The choice (4.71), as it becomes clear in the previous program listing, implies a *random update*. At each time, the variable to be updated is chosen randomly among the set of N variables. It is possible to use the so-called sequential update in which the variables are chosen one after the other: first s_1, next s_2, then s_3, and so on. From the practical point of view, it simply means the replacement of the line `i=i_ran()` by `i=k`, or removal of that line and changing the loop variable to `do i=1,n`. There are several advantages to this procedure. First, it is faster since it avoids calling the integer random number routine. Second, it makes sure that in every Monte Carlo step each and every variable has been selected for updating. Note

that, in random update, the probability that one variable has not been chosen after 1 MCS (N individual proposals) is $(1-1/N)^N$, which tends to $e^{-1}=0.368$ for large N. The most notorious implication is that the Markov chain is not homogeneous because the transition probability $f_{\hat{\mathbf{x}}_n}(x'|x)$ depends now on n. In a sequential update at the first trial, we chose variable s_1 at second, s_2, and so on up to the $N+1$ trial, where we choose again $s_{N+1}=s_1$. This is, such that at trial n we chose variable s_{i_n} with $i_n=(n-1 \bmod N)+1$ and the transition probability $f_{\hat{\mathbf{x}}_n}(x'|x)=f_{i_n}(x'|x)$ depends on the number i_n only. As before, once we have chosen the variable s_{i_n} for updating, the proposal and acceptance probabilities are given by the corresponding expressions $g_{i_n}(x|x')$ and acceptance probabilities $h_{i_n}(x'|x)$ which now depend on i_n. The evolution of the pdf for this nonhomogeneous process is

$$\begin{aligned} f_{\hat{\mathbf{x}}_{n+1}}(x) &= \int f_{i_n}(x|y) f_{\hat{\mathbf{x}}_n}(y)\, dy \\ &= \int h_{i_n}(x|y) g_{i_n}(x|y) f_{\hat{\mathbf{x}}_n}(y)\, dy + f_{\hat{\mathbf{x}}_n}(x)\left[1-\int dz h_{i_n}(z|x) g_{i_n}(z|x)\right]. \end{aligned} \tag{4.74}$$

As we chose $g_{i_n}(x|x')$ and $h_{i_n}(x'|x)$ to satisfy the detailed balance condition $g_{i_n}(x|x')h_{i_n}(x'|x)f_{\hat{\mathbf{x}}}(x)=g_{i_n}(x'|x)h_{i_n}(x|x')f_{\hat{\mathbf{x}}}(x')$ for each value of i_n, then it is easy to prove that the distribution $f_{\hat{\mathbf{x}}}(x)$ is still a stationary distribution for (4.74). We need only to care about ergodicity and make sure that the sequential update allows, in principle, any possible values for the variables to be reached to make sure that $f_{\hat{\mathbf{x}}}(x)$ will be asymptotically sampled by the nonhomogeneous Markov chain.

4.6 Heat-Bath Method

At variance with the previous methods, heat bath is the first one that requires the variable to sample to be N-dimensional $\hat{\mathbf{x}}=(\hat{\mathbf{s}}_1,\ldots,\hat{\mathbf{s}}_N)$ with $N>1$. The idea is to solve the detailed balance condition (4.38) by (i) proposing transitions $x\to x'$ in which only one variable changes, $s_i\to s_i'$, and (ii) taking an acceptance probability equal to 1, $h(x'|x)=1$. The detailed balance condition is reduced to

$$g(x'|x)f_{\hat{\mathbf{x}}}(x)=g(x|x')f_{\hat{\mathbf{x}}}(x') \tag{4.75}$$

with $x=(s_1,\ldots,s_i,\ldots,s_N)$ and $x'=(s_1,\ldots,s_i',\ldots,s_N)$. This equation is satisfied if we take

$$g(x'|x)=f(s_i'|s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_N). \tag{4.76}$$

This is easily checked by replacing

$$g(x'|x)=f(s_i'|s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_N)=\frac{f_{\hat{\mathbf{x}}}(s_1,\ldots,s_{i-1},s_i',s_{i+1},\ldots,s_N)}{f(s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_N)} \tag{4.77}$$

and

$$\begin{aligned} g(x|x') &= f(s_i|s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_N) \\ &= \frac{f_{\hat{\mathbf{x}}}(s_1,\ldots,s_{i-1},s_i,s_{i+1},\ldots,s_N)}{f(s_1,\ldots,s_{i-1},s_{i+1},\ldots,s_N)} \end{aligned} \tag{4.78}$$

in (4.75).

In words, the heat-bath algorithm proceeds as follows:

1) Chose a variable s_i. This step can be done either randomly or sequentially.
2) Sample a new value s_i' from a conditional distribution in which all other variables $(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N)$ are fixed. The proposed value is always accepted. We will see examples of the use of this algorithm when we discuss in Chapter 5 some applications to systems of interest in statistical mechanics.

4.7 Tuning the Algorithms

4.7.1 Parameter Tuning

In a Monte Carlo simulation, it is very clear what we mean by the optimal algorithm: the one that gives the smallest statistical error using the same computer time. If we remember that the error of the estimator is $\sigma[\hat{\boldsymbol{\mu}}_M]$ as given by (4.24), besides increasing the number of samples M (which obviously increases the computer time), one should try to reduce the correlation time τ_G (without increasing the computer time). For instance, when choosing the proposal probability (4.57) for the Gaussian distribution (but similar arguments apply to other cases), Δ is a parameter that can be varied without changing the computer time. It is clear that τ_G is a function of Δ. If Δ is very small, the proposal x' is very close to the actual value x and will be accepted very often. A series of numbers G_k in which G_{k+1} is very close to G_k has a large correlation time. Similarly, when Δ is very large, the acceptance probability is very small and most values are rejected. As a consequence, in many occasions, it is $G_{k+1} = G_k$ and, again, the correlation time is very large. Is expected, then, that there exists an optimal value of Δ that yields the minimum correlation time τ_G. Unfortunately, it is usually not so easy to determine with precision the optimal value Δ. Analytically, the problem is very difficult. To know τ_G, according to (4.19) and (4.23), we need to know the correlation function $\langle G_k G_j \rangle$, which is an average in the stationary distribution. This is difficult as the iteration equation (4.42) can rarely be solved explicitly. It is possible to obtain an expression for the one-step correlation using

$$\begin{aligned}\langle G_{n+1} G_n \rangle &= \int dx\ dy\, G(x) G(y) f_{\hat{\mathbf{x}}_{n+1}, \hat{\mathbf{x}}_n}(x, y) \\ &= \int dx\ dy\, G(x) G(y) f(x|y) f_{\hat{\mathbf{x}}_n}(y). \end{aligned} \tag{4.79}$$

Replacing $f(x|y)$ from (4.36) and $f_{\hat{\mathbf{x}}_n}(y)$ for the stationary distribution $f_{\hat{\mathbf{x}}}(y)$, one obtains after some straightforward algebra that the one-time normalized correlation function is

$$\begin{aligned}\rho_G(1) &= \frac{\langle G_{n+1} G_n \rangle - \langle G_k \rangle^2}{\sigma^2[G]} \\ &= 1 - \frac{1}{2\sigma^2[G]} \int dx\ dy\, h(x|y) g(x|y) [G(x) - G(y)]^2 f_{\hat{\mathbf{x}}}(y). \end{aligned} \tag{4.80}$$

To obtain from here the correlation time, we make the approximation that the correlation function decays exponentially $\rho_G(i) \approx [\rho_G(1)]^i$ and use (4.26). Although this approximation may not necessarily be accurate, it does give us an estimate for the correlation time τ_G.

For the Metropolis *et al.* algorithm applied to the Gaussian distribution, it is possible to perform analytically the integrals (4.80) to obtain, after a tedious calculation, that for $G(x) = x$ the one-time correlation function is

$$\rho_x(1) = 1 - \frac{\Delta^2}{6}\left[1 - \mathrm{erf}\left(\frac{\Delta}{2\sqrt{2}}\right)\right] + \frac{16}{3\Delta\sqrt{2\pi}}\left[\left(\frac{\Delta^2}{8} + 1\right)e^{-\Delta^2/8} - 1\right] \quad (4.81)$$

where we can check that it has a minimum value at $\Delta \approx 3.70$, which is hence the optimal value for this parameter.

In general, it will not be possible to perform the integrals (4.80), but one can always compute the one-time correlation function $\rho_G(1)$ from the numerical algorithm and estimate numerically the correlation time τ_G using the approximation (4.26). This is relatively easy to do. Alternatively, one could compute numerically the whole correlation function $\rho_G(k)$ and derive the correlation time from there. We have left for Appendix C the explanation of an efficient algorithm to compute the correlation time of a series.

If all the above seems too complicated (although it should not really be), there is another strategy to determine the optimal value of the tunable parameters. We have already commented on the fact that very large or very small acceptance probabilities lead to large correlation times. The simple answer then is to choose parameters such that the average acceptance probability is neither very large nor very small, which, for a number ϵ bounded to be between 0 and 1, means, roughly speaking, $\epsilon = 1/2$. This is a heuristic rule, with not much justification, but to choose the parameters using this criterion is usually better than not tuning parameters at all. For instance, at the optimal value $\Delta = 3.70$ for the Gaussian distribution (the minimum of the one-time correlation function), the acceptance probability (as given by (4.61)) is $\epsilon = 0.418$, not equal to $1/2$, but not too far from it either.

4.7.2 How Often?

The computer time spent in the calculation of the estimator $\hat{\boldsymbol{\mu}}_M[G]$ to the integral of the function $G(x)$ can be divided into the time necessary, say t_2, to generate one value x_k of the random variable and the time, say t_1, it takes to compute the function $G(x_k)$ and make the necessary sums, products, and sums of squares to compute the estimator, the error, the correlation function, and so on. If the correlation time τ_G is small (of the order of 1), then there is little correlation between all values of $G(x_k)$ and including them all in the averages makes perfect sense. Imagine, however (and, alas! this occurs more often that one desires), that the correlation time τ_G is large. Then there is a large correlation between the different values of $G(x_k)$ and it might not be useful to include all of them in the calculation of the averages. This means to let pass a number of proposal/acceptance steps, say K,

between measurements. In this way, the new decimated series of values $G(x_k)$ has a correlation time τ_G/K. Is there an optimal value for K? The answer is yes, and a simple calculation of its optimal value can save us a lot of computer time in complicated problems, where the calculation of $G(x_k)$ is time consuming.

So, let us assume that we make M measurements and that there are K proposal/acceptance steps between measurements. The correlation time is τ_G/K, and the variance of the estimator using these values is, according to (4.24)

$$\frac{\sigma^2[\hat{\mathbf{G}}]}{M}\left(2\frac{\tau_G}{K}+1\right) \tag{4.82}$$

and we would like to choose K and M to minimize this expression. The minimization has to be performed under the condition that the total computer time is constant, which is $T_2 = KMt_2$ (time needed to generate the KM steps) plus $T_1 = Mt_1$ (measurement time). The minimization of

$$\frac{1}{M}\left(2\frac{\tau_G}{K}+1\right) \tag{4.83}$$

with the constraint

$$Mt_1 + KMt_2 = t \text{ constant} \tag{4.84}$$

yields

$$K = \sqrt{\frac{2\tau_G t_1}{t_2}} \tag{4.85}$$

as the optimal value of K. The ratio of the time spent in measuring to the time spent in updating is

$$\frac{T_1}{T_2} = \frac{Mt_1}{KMt_2} = \sqrt{\frac{t_1}{2\tau_G t_2}}. \tag{4.86}$$

This time there does not seem to be an easy heuristic rule to replace this calculation. Sometimes it is used as the criterion of choosing K such that the ratio of measuring to updating is $T_1/T_2 = 1$ or $K = t_1/t_2$, a criterion not sustained by this simple calculation.

4.7.3 Thermalization

We reach now a delicate point. As mentioned in Section 1.8, the use by dynamical methods of the adequate homogeneous Markov chain, with the assumption of ergodicity, ensures that the values of the random variable $\hat{\mathbf{x}}$ from a pdf $f_{\hat{\mathbf{x}}}(x)$ are reached *asymptotically*. Namely,

$$\lim_{n\to\infty} f_{\hat{\mathbf{x}}_n}(x) = f_{\hat{\mathbf{x}}}(x). \tag{4.87}$$

According to this criterion, we should wait for an ''infinite'' number of steps before reaching the stationary state and start the measurements.

We have already mentioned that, in practice, "$n \to \infty$" means that we have to discard the first M_0 steps of the algorithm, that is, the first M_0 values of the random variable $\hat{\mathbf{x}}$. The problem is to determine faithfully the value of M_0. We can affirm that underestimating the value of M_0 is the main source of uncontrolled mistakes in the field of Monte Carlo simulations and there are many published papers that are wrong just because the authors did not wait long enough to reach the stationary state of the Markov chain.

The process of discarding the first M_0 steps is called *thermalization*. So, the key question is, how long should we thermalize? How to get an estimate of M_0? Obviously, one thing to do is to vary M_0 and check that the results, within the unavoidable statistical errors, do not depend on M_0. We can be a little bit more precise and compute the time dependence of the averages. For that, we need to compute the so-called non-linear relaxation function ρ_G^{NL} defined as

$$\rho_G^{\mathrm{NL}}(n) = \frac{\langle G_n \rangle - \langle G \rangle_{\mathrm{st}}}{\langle G_0 \rangle - \langle G \rangle_{\mathrm{st}}}. \tag{4.88}$$

Here

$$\langle G_n \rangle = \int_{-\infty}^{\infty} f_{\hat{\mathbf{x}}_n}(x) G(x)\, dx \tag{4.89}$$

denotes an average using the different values that are generated at step n using a different value from the initial pdf $f_{\hat{\mathbf{x}}_0}(x)$, and $\langle G \rangle_{\mathrm{st}}$ is the stationary value. The thermalization time is related to the time it takes for $\rho_G^{\mathrm{NL}}(n)$ to decay from $\rho_G^{\mathrm{NL}}(n) = 1$ to its stationary value $\rho_G^{\mathrm{NL}}(n) = 0$. This can be estimated similar to the correlation function by introducing the non-linear correlation time

$$\tau_G^{\mathrm{NL}} = \sum_{n=0}^{\infty} \rho_G^{\mathrm{NL}}(n) \tag{4.90}$$

and, finally, taking M_0 as a number of times τ_G^{NL}, for example, $M_0 \geq 10\tau_G^{\mathrm{NL}}$. It is clear that the problem in this procedure is that the definition (4.88) depends on the stationary value $\langle G \rangle_{\mathrm{st}}$, which can be determined free of systematic errors only if we are sure that we are in the stationary state. There are examples (mostly in the field of phase transitions, but not only there) in which the decay is very slow and it is difficult to determine whether the system has reached indeed the stationary state.

There is one check, however, that can help us in this issue. There is a simple relation

$$\langle q(y|x) \rangle_{\mathrm{st}} = 1 \tag{4.91}$$

that holds in the stationary state. The averaging has to be performed with respect to the *proposed* values y (hence, distributed according to $g(y|x)$) over the allegedly stationary distribution of x. Namely

$$\langle q(y|x) \rangle_{\mathrm{st}} = \int dy \int dx f_{\hat{\mathbf{x}}}(x) g(y|x) q(y|x). \tag{4.92}$$

The proof is simple, replace $q(x|y)$ by its definition and manipulate

$$\begin{aligned}\langle q(y|x)\rangle_{\rm st} &= \int dy \int dx f_{\hat{\mathbf{x}}}(x) g(y|x) q(y|x) \\ &= \int dy \int dx f_{\hat{\mathbf{x}}}(x) g(y|x) \frac{g(x|y) f_{\hat{\mathbf{x}}}(y)}{g(y|x) f_{\hat{\mathbf{x}}}(x)} \\ &= \int dy \left[\int dx g(x|y) \right] f_{\hat{\mathbf{x}}}(y) \\ &= \int dy f_{\hat{\mathbf{x}}}(y) = 1.\end{aligned} \tag{4.93}$$

Therefore, if we are in the stationary state, we should verify (4.91). Unfortunately, this is a necessary condition, but not a sufficient one and we can get values of $\langle q(x|y)\rangle_{\rm st}$ close to 1 and still not be in the stationary state. In practice, we should get a value of $\langle q(x|y)\rangle_{\rm st}$ really close to 1, say a zero followed by a good number of 9's (at least three or four). Otherwise, we most probably have not yet reached the stationary state.

Further Reading and References

The Metropolis *et al.* method first appeared in ref. [9]. The heat-bath method was introduced by Creutz [10] in his study of lattice gauge theories. Glauber acceptance probability was introduced in [11] in a classical work of dynamics in simple Ising chain. van Beijeren and Schulman acceptance probability was introduced in [12]. Examples of the use of the non-linear correlation function to study the approach to the steady state are given in the book by D.P. Landau and K. Binder [14].

Exercises

1) Sample the N-dimensional distribution (4.30) using a simple rejection method with $(x_1, \ldots, x_N)$ independent and uniformly distributed in the interval $[-1, 1]$ and check that the average acceptance probability decays exponentially with N.
2) Prove that in a homogeneous Markov chain, the correlation function $\rho(i, j)$ depends only on the difference $|i - j|$.
3) Use (4.44) with the proposal (4.57) and the Metropolis *et al.* choice $h(x|y) = \min(1, q(x|y))$, to reproduce the expression (4.61). Check the correctness of this result by comparing with numerical simulations by varying the parameter Δ. Compare (numerically) with the average acceptance probability ϵ obtained using the Glauber choice $h(x|y) = \dfrac{q(x|y)}{1 + q(x|y)}$. Which method has the largest acceptance probability?
4) Continuing with the previous problem, compute numerically, both for Metropolis *et al.* and Glauber acceptance probabilities, the correlation time in the stationary state for the function $G(x) = x^2$ as a function of the parameter

Δ and determine the optimal values of Δ. Conclude which choice is more efficient (do not forget to take into consideration the computer time needed by each algorithm).

5) Repeat the previous problem using the function $G(x) = \cos(x)$ and check that the numerical value obtained for the integral

$$\int_{-\infty}^{\infty} dx \cos(x) \exp\left(-\frac{x^2}{2}\right)$$

using the Metropolis *et al.* algorithm for both choices of the acceptance probability $h(x|y)$ agrees within errors with the exact result $\sqrt{2\pi/e}$.

6) Compute as a function of λ the correlation time of the Metropolis *et al.* algorithm applied to the generation of the Poisson distribution of the parameter λ. Take $\lambda = 0.5, 1.0, 2.0, 5.0, 10.0, 100.0$.

7) Program the Metropolis algorithm for the generation of the Poisson distribution using a proposal $g(k'|k)$ uniformly distributed in the interval $(k - L, k + L)$. Compute the correlation time τ_k associated to the function $G(k) = k$ and find the value of L that makes τ_G minimum and discuss its dependence on the parameter λ of the Poisson distribution.

8) Consider a three-dimensional discrete random variable $\hat{\mathbf{x}}$ that can take values in $\Omega = \{-1, 1\}^3$. In other words, the possible values are: $x_1 = (1, 1, 1)$, $x_2 = (1, 1, -1)$, $x_3 = (1, -1, 1)$, $x_4 = (1, -1, -1)$, $x_5 = (-1, 1, 1)$, $x_6 = (-1, 1, -1)$, $x_7 = (-1, -1, 1)$, and $x_8 = (-1, -1, -1)$. Writing $x_i = (s_1^i, s_2^i, s_3^i)$, we assign to each possible value x_i a probability p_i given by

$$p_i = C \exp[K(s_1^i s_2^i + s_2^i s_3^i + s_3^i s_1^i)]$$

where K is a given number and C is the normalization constant. Write programs that implement Metropolis *et al.* algorithm with the (i) Metropolis *et al.*, (ii) Glauber, and (iii) van Beijeren and Schulman choices for the acceptance probability to generate values of $\hat{\mathbf{x}}$ distributed according to these probabilities. Use proposal probabilities (i) in which only one of the s_i variables is proposed for change (which is the only possible proposal?) and (ii) in which the proposed value is (s_1', s_2', s_3') is randomly chosen from the 8 possible values.

9) Given the functions $M(x) = (s_1 + s_2 + s_3)/3$ and $E(x) = (s_1 s_2 + s_2 s_3 + s_3 s_1)/3$ defined on space Ω of the previous problem, compute the linear and nonlinear correlation times of the Metropolis *et al.* algorithm. Use this result to compute $\langle M \rangle$, $\langle M^2 \rangle$, $\langle E \rangle$, $\langle E^2 \rangle$ as a function of K (take $K = 0.25, 0.5, \ldots, 4.0$). Compare with the exact results obtained computing the averages summing over the eight possible values of the variable $\hat{\mathbf{x}}$.

10) Repeat the two previous exercises for the space $\Omega = \{-1, 1\}^{1000}$. Which is the only proposal that gives a nonvanishing acceptance probability now?

11) For the Gaussian generator that uses the Metropolis algorithm for different values of the parameter $\Delta \in (1, 10)$, generate long series $(x_1, \ldots, x_M)$ and compute the correlation function and correlation time of x_i using, for instance, the routine explained in Appendix C. Determine the value of Δ for which the correlation time is minimum.

12) Consider the Metropolis *et al.* method applied to the distribution (4.70). Use a proposal $g(y|x)$ in which *all* variables s_i, $i = 1, \ldots, N$ are proposed a new value in the interval $(s_i - \Delta, s_i + \Delta)$. Determine the dependence of Δ with the number of variables N such that the acceptance probability is constant and close to 0.5. Using this value, compute the correlation time as a function of N. Compare its value with the one obtained by the method in which *only one* variable is chosen for updating at a given step and Δ does not depend on N. Which method is more efficient?

13) Check numerically that $\langle q(x|y)\rangle_{\text{st}} = 1$ for both variants of the Metropolis *et al.* method used in the previous exercise.

14) Prove that K given by (4.85) is the optimal choice for the number of updates between measurements in order to minimize the computer time.

15) For the rejection with repetition method explained in Section 4.1, compute analytically the correlation function $C_{\text{st}}(n) = \dfrac{\langle \hat{\mathbf{x}}_n \hat{\mathbf{x}}_0 \rangle - \langle \hat{\mathbf{x}}_n \rangle \langle \hat{\mathbf{x}}_0 \rangle}{\sigma^2[\hat{\mathbf{x}}_0]}$, in the steady state, as well as the non-linear correlation function

$$C_{\text{nl}}(n) = \frac{\langle \hat{\mathbf{x}}_n \rangle - \langle \hat{\mathbf{x}}_\infty \rangle}{\langle \hat{\mathbf{x}}_0 \rangle - \langle \hat{\mathbf{x}}_\infty \rangle}.$$

5
Applications to Statistical Mechanics

5.1
Introduction

One of the most important applications of the Monte Carlo sampling techniques appears in the field of statistical mechanics, which deals with systems with a large number of degrees of freedom N. A system is described by a set of coordinates $X = (x_1, \ldots, x_N)$. These variables x_i are typically positions, or angles, but more complicated examples of generalized coordinates exist in the literature. Each coordinate x_i has an associated conjugate momentum p_i and the whole set of X and $P = (p_1, \ldots, p_N)$ variables is required to fully specify the state of the system. We will denote by $\Gamma = (X, P)$ the combined set of $2N$ coordinate and momentum variables and a point in the Γ ''phase space'' as a ''microscopic configuration.'' It is essential to remember that the simplest macroscopic system will be described by an enormously large number of variables N. An order of magnitude of how large this number can be in typical situations is Avogadro's number $N_A = 6.022 \times 10^{23}$, the number of molecules in a mol, usually a few grams, of a substance.

A very important function is the Hamiltonian $\mathcal{H}(X, P) = \mathcal{H}(x_1, \ldots, x_N, p_1, \ldots, p_N)$. It is important in many senses. First of all, it determines, via Hamilton's equations

$$\frac{dx_i}{dt} = \frac{\partial \mathcal{H}}{\partial p_i}, \tag{5.1}$$

$$\frac{dp_i}{dt} = -\frac{\partial \mathcal{H}}{\partial x_i}, \qquad i = 1, \ldots, N \tag{5.2}$$

the time evolution of $x_i(t), p_i(t), i = 1, \ldots, N$. All we need to do is to solve this set of $2N$ differential equations given some initial condition at, say, time $t = 0$, which, of course, is a tremendous (and impossible) task in most cases.

Second, and more important to us, is that the Hamiltonian can also be used to determine the probability density function of observing at thermal equilibrium some set of values for the coordinates and momenta $\Gamma = (X, P)$: that is, the probability of a microscopic configuration. Boltzmann and Gibbs showed that such a pdf is given by what is nowadays called the *Boltzmann factor*, $e^{-\beta \mathcal{H}}$, in the following manner:

$$f(\Gamma) = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}} \tag{5.3}$$

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

where

$$\mathcal{Z} = \int d\Gamma\, e^{-\beta \mathcal{H}} \tag{5.4}$$

is the normalization factor $d\Gamma$ is the volume element in phase space and $\beta = 1/kT$ is the inverse of the temperature rescaled by Boltzmann's constant k. Just not to hide anything, we can write this normalization factor in full, as

$$\mathcal{Z} = \int dx_1 \dots dx_N\, dp_1 \dots dp_N\, e^{-\beta \mathcal{H}(x_1,\dots,x_N,p_1,\dots,p_N)}. \tag{5.5}$$

In fact, this normalization factor by itself is very important, so important that it has a name, ''the partition function.'' The reason of its importance is that the probability of a configuration is something difficult to determine experimentally, while typical measures concern the so-called thermodynamic potentials: Helmholtz free energy F, internal energy $\mathcal{U}$, enthalpy H, or the equation of state giving the pressure P as a function of volume V and temperature, or the specific heat at constant volume C_V, or the isothermal compressibility κ_T, and so on. The framework of statistical mechanics tells us that all these quantities can be derived from the knowledge of the partition function as a function of only the volume V, the number of particles $\mathbf{N}$[1)], and the temperature T. For example, Helmholtz's free energy is given by

$$F(\mathbf{N}, V, T) = -kT \log \mathcal{Z}, \tag{5.6}$$

the internal energy is

$$\mathcal{U} = \left(\frac{\partial (F/T)}{\partial (1/T)} \right)_{V,\mathbf{N}}, \tag{5.7}$$

That is, the variation of F/T with respect to $1/T$ when both N and V are kept constant. Specific heat at constant volume is

$$C_V = \left(\frac{\partial \mathcal{U}}{\partial T} \right)_{V,\mathbf{N}}. \tag{5.8}$$

The entropy can be computed from $S = (\mathcal{U} - F)/T$ or directly, as

$$S = -\left(\frac{\partial F}{\partial T} \right)_{V,\mathbf{N}} \tag{5.9}$$

and so on. This ''recipe'' of statistical mechanics is extremely difficult to carry out in practice because the integrals involved in the definition of the partition function (5.5) can only be performed for a limited number of simple examples: a gas of non-interacting particles, a system of independent harmonic oscillators, and so on, and a limited number of not-so-simple examples, the Gaussian free model and the Ising model for ferromagnetism being the most notable ones. Furthermore, the framework of statistical mechanics shows that some interesting macroscopic observable phenomena only occur in the limit of the number of degrees of freedom N tending to infinity, which makes the calculation of the partition function usually even harder.

1) The number of particles $\mathbf{N}$ is not the same as the number of degrees of freedom N, as each particle can have, in general, more than one degree of freedom.

An alternative approach is to work not with the partition function directly but with the pdf of the microscopic configurations, $f(\Gamma)$. In this approach, some observables are defined in terms of averages with respect to this pdf. The average of any function of the microscopic variables $G(\Gamma)$ is defined in the usual way, as

$$\langle G \rangle = \int d\Gamma G(\Gamma) f(\Gamma). \tag{5.10}$$

There are many examples. The internal energy can be computed as the average value of the Hamiltonian

$$\mathcal{U} = \langle \mathcal{H} \rangle. \tag{5.11}$$

The specific heat, $C_V = \left(\frac{\partial \mathcal{U}}{\partial T} \right)_{\mathbf{N},V}$, can also be obtained using higher order moments of the Hamiltonian:

$$C_V = k\beta^2 \left[\langle \mathcal{H}^2 \rangle - \langle \mathcal{H} \rangle^2 \right] \tag{5.12}$$

and so on. Unfortunately, not all relevant magnitudes can be computed this way. For instance, the entropy S cannot be interpreted as the average value of some known function G[2)].

Many problems of statistical mechanics in equilibrium can then be reduced to the calculation of averages using the pdf (5.3). This probabilistic description (which lies at the core of statistical mechanics) is, in some cases, irrespective of whether the dynamical variables themselves satisfy Hamilton's equations. For instance, in problems of magnetism, a very successful approach consists in considering that the variables x_i represent microscopic magnetic moments that interact among themselves. In the simplest version, known as the *Ising model*, these microscopic variables can take only two possible values $x_i \equiv \mu s_i$, with μ being the unit of magnetic moment and $s_i = \pm 1$ is a rescaled variable (the Ising or spin variable). There are no momenta-like variables p_i associated with these magnetic moment variables and there are no Hamilton's equations. It does not even make sense to compute the time derivative of a noncontinuous variable s_i that can only take two possible values. Still, the microscopic variables s_i interact via the so-called Hamiltonian function $\mathcal{H}(s_1, \ldots, s_N)$ and the probability of observing a particular configuration $S = (s_1, \ldots, s_N)$ is $f(S) = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}}$. The partition function now is not the integral over all values of s_i but, as s_i can take only two values $s_i = \pm 1$, it is computed as a sum

$$\mathcal{Z} = \sum_{s_1 = \pm 1} \cdots \sum_{s_N = \pm 1} e^{-\beta \mathcal{H}}. \tag{5.13}$$

The Hamiltonian $\mathcal{H}$ takes into account the magnetic interactions, and it is typically simplified in order to consider only the interactions that occur between these Ising

2) It is indeed possible to derive the formula $S = -\int d\Gamma f(\Gamma) \log[f(\Gamma)] + C$, with C a constant. Though formally this could be thought of as the average $S = -\langle \log f \rangle + C$, the truth is that, in order to perform this averaging, we must know the pdf $f(\Gamma)$, *including the normalization constant* $\mathcal{Z}$, which is usually impossible. If we knew the partition function, we would not need any further integrals in order to compute the entropy; we would simply use (5.6–5.9).

variables s_i which are close in space, neglecting the magnetic interactions with variables that are farther apart than some minimum cutoff distance. We will see specific examples later.

Once we have reduced the problem of equilibrium statistical mechanics to the calculation of averages, it is clear which should be an efficient numerical approach: replace the true average (5.10) by a sample average

$$\langle G \rangle = \mu[G] \pm \frac{\sigma[G]}{\sqrt{M}}(2\tau_G + 1) \tag{5.14}$$

with

$$\mu[G] = \frac{1}{M}\sum_{k=1}^{M} G(\Gamma_k), \tag{5.15}$$

$$\sigma^2[G] = \frac{1}{M}\sum_{k=1}^{M} G(\Gamma_k)^2 - (\mu[G])^2 \tag{5.16}$$

with Γ_k, $k = 1, \ldots, M$ being the set of generated configurations and τ_G the associated autocorrelation time of G obtained from the autocorrelation function $\rho_G(i)$. The key point is now the generation of characteristic configurations Γ_k distributed according to $f(\Gamma)$ in a problem with many variables. But this is precisely what we have claimed that Monte Carlo algorithms are good at! We now rephrase the dynamical methods of last chapter within the framework of statistical mechanics problems.

In the dynamical methods, we proposed a change from a configuration Γ to a configuration Γ' taken from a pdf $g(\Gamma'|\Gamma)$. This proposal was then accepted with probability $h(\Gamma'|\Gamma)$. In order to ensure that the stationary distribution is $f(\Gamma)$, it is sufficient to demand that these conditional functions satisfy the detailed balance condition

$$g(\Gamma'|\Gamma)h(\Gamma'|\Gamma)f(\Gamma) = g(\Gamma|\Gamma')h(\Gamma|\Gamma')f(\Gamma') \tag{5.17}$$

with $f(\Gamma)$ now given by (5.3). A solution to this functional equation is given by the Metropolis algorithm, in which a proposal probability $g(\Gamma'|\Gamma)$ is first selected and then the acceptance probability is

$$h(\Gamma'|\Gamma) = \min\left[1, q(\Gamma'|\Gamma)\right] \tag{5.18}$$

where

$$q(\Gamma'|\Gamma) = \frac{g(\Gamma|\Gamma')f(\Gamma')}{g(\Gamma'|\Gamma)f(\Gamma)}. \tag{5.19}$$

Most algorithms[3)] assume that $g(\Gamma'|\Gamma)$ is a symmetric function, $g(\Gamma'|\Gamma) = g(\Gamma|\Gamma')$. In this case, and after replacing the expression for $f(\Gamma)$, we obtain

$$q(\Gamma'|\Gamma) = \frac{\mathcal{Z}^{-1}e^{-\beta\mathcal{H}(\Gamma')}}{\mathcal{Z}^{-1}e^{-\beta\mathcal{H}(\Gamma)}} = e^{-\beta\Delta\mathcal{H}} \tag{5.20}$$

3) But not all. A notable exception is the hybrid Monte Carlo algorithm explained in Chapter 10.

with $\Delta\mathcal{H} = \mathcal{H}(\Gamma') - \mathcal{H}(\Gamma)$, the change in energy involved in the proposal $\Gamma \to \Gamma'$. Note that the partition function $\mathcal{Z}$ disappears from the expression for $q(\Gamma'|\Gamma)$. The acceptance probability becomes

$$h(\Gamma'|\Gamma) = \min(1, e^{-\beta\Delta\mathcal{H}}). \tag{5.21}$$

This is the original proposal of the celebrated paper by Metropolis *et al.* [9]. It has an intuitive physical interpretation. The characteristic configurations at equilibrium are those that minimize Helmholtz's free energy $F = \mathcal{U} - TS$, a balance between the internal energy $\mathcal{U}$ (which tends to a minimum) and entropy S (which tends to a maximum). As $\Delta\mathcal{H} \leq 0$ implies $h(\Gamma'|\Gamma) = 1$, this balance is achieved by (i) accepting all proposals $\Gamma \to \Gamma'$ in which the energy is reduced, and (ii) accepting those proposals in which energy increases $\Delta\mathcal{H} > 0$ with a probability $e^{-\beta\Delta\mathcal{H}}$. As $\beta = 1/kT$, when $T \to 0$ the probability of accepting a proposal that increases the energy tends to 0. On the contrary, when $T \to \infty$, it is $\beta \to 0$, the acceptance probability tends to 1, and every proposal is accepted independently of the energy cost.

Another solution to the detailed balance condition is that of Glauber:

$$h(\Gamma'|\Gamma) = \frac{q(\Gamma'|\Gamma)}{1 + q(\Gamma'|\Gamma)}, \tag{5.22}$$

or, using (5.20),

$$h(\Gamma'|\Gamma) = \frac{1}{1 + e^{\beta\Delta\mathcal{H}}}. \tag{5.23}$$

But other choices for the acceptance probability h are still possible, as discussed in Section 4.4 of Chapter 4.

5.2
Average Acceptance Probability

We now rewrite (4.91) in order to derive the average acceptance probability in the case that the pdf is the exponential of a Hamiltonian: $f_{\hat{\mathbf{x}}}(\Gamma) = \mathcal{Z}^{-1}e^{-\beta\mathcal{H}(\Gamma)}$, and assuming a symmetric proposal $g(\Gamma|\Gamma') = g(\Gamma'|\Gamma)$.

$$\left\langle e^{-\beta\Delta\mathcal{H}} \right\rangle_{\text{st}} = 1 \tag{5.24}$$

where $\Delta\mathcal{H} = \mathcal{H}(\Gamma') - \mathcal{H}(\Gamma)$ is the change of energy involved in the proposal $\Gamma \to \Gamma'$. If we use now Jensen's inequality $\langle e^{-z} \rangle \geq e^{-\langle z \rangle}$, which is valid for any random variable z, with $z = \beta\Delta\mathcal{H}$, we derive that, in the steady state, the average value of the proposed changes of energy is always greater than zero, $\langle \Delta\mathcal{H} \rangle > 0$.

Although it is possible to be more general, we present here a simplified treatment which assumes that the distribution of the proposed changes of energy $z = \beta\Delta\mathcal{H}$ can be well approximated by a Gaussian distribution of average μ and variance σ^2. As it is known that for a Gaussian distribution it is $\langle e^{-z} \rangle = e^{-\mu+\sigma^2/2}$ and we have proved that $\langle e^{-z} \rangle = 1$, we derive that this Gaussian assumption is consistent if we take $\sigma^2 = 2\mu$. Let us now compute the average value of the Metropolis acceptance

probability $h(\Gamma'|\Gamma) = \min[1, e^{-\beta\Delta\mathcal{H}}]$. Under the assumption that $z = \beta\Delta\mathcal{H}$ follows a Gaussian distribution, we obtain for the average value

$$\langle h(\Gamma'|\Gamma)\rangle_{\text{st}} = \int_{-\infty}^{\infty} \mathrm{d}z \min[1, e^{-z}] \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(z-\mu)^2}{2\sigma^2}}. \tag{5.25}$$

Performing the integral and replacing $\sigma^2 = 2\mu$, we obtain the simple result $\langle h(\Gamma'|\Gamma)\rangle_{\text{st}} = \text{erfc}\left(\frac{\sqrt{\mu}}{2}\right)$ or

$$\langle h(\Gamma'|\Gamma)\rangle_{\text{st}} = \text{erfc}\left(\frac{\sqrt{\beta\langle\Delta\mathcal{H}\rangle}}{2}\right) \tag{5.26}$$

where $\text{erfc}(z) = 1 - \text{erf}(z)$ is the complementary error function. It is worth noting that, in the limit of large $\langle\Delta\mathcal{H}\rangle$, we can use $\text{erfc}(z) \underset{z\to\infty}{\longrightarrow} \frac{e^{-z^2}}{z\sqrt{\pi}}$ [4] to obtain the asymptotic result

$$\langle h(\Gamma'|\Gamma)\rangle_{\text{st}} \longrightarrow \frac{2e^{-\frac{\beta\langle\Delta\mathcal{H}\rangle}{4}}}{\sqrt{\pi\beta\langle\Delta\mathcal{H}\rangle}} \tag{5.27}$$

which is valid for $\Delta\mathcal{H} \to \infty$ and shows that the average acceptance probability $\langle h(\Gamma'|\Gamma)\rangle$ goes to zero *exponentially* with the average (positive) energy change $\langle\Delta\mathcal{H}\rangle$. As $\mathcal{H}$ is usually an extensive quantity proportional to the number of degrees of freedom N, it is important to devise proposals in which the change of energy $\Delta\mathcal{H}$ can be kept small, $O(1)$ instead of $O(N)$; otherwise, the small average acceptance probability will yield a very large correlation time and the corresponding large statistical errors will make the estimator useless. The standard trick to keep a reasonable acceptance probability is to propose new configurations Γ' in which only a few degrees of freedom have been modified from configuration Γ. Collective updates in which all degrees of freedom change at once but still the acceptance probability is nonvanishingly small are much more difficult to devise. We will devote Chapter 10 and Appendix D to explain some collective update algorithms, and restrict ourselves in the rest of this chapter to algorithms in which the proposal involves the change of a few (maybe only one) variable.

Let us know explain some applications of the Monte Carlo algorithms to the study of particular systems of interest in statistical mechanics.

5.3 Interacting Particles

The first case we will consider is that of interacting particles without internal degrees of freedom. This means that all we need to specify the state are the spatial locations $\vec{r}_i$ of the $i = 1, \ldots, \mathbf{N}$ particles, as well as the associated momenta $\vec{p}_i = m\vec{v}_i$, with m being the mass of the particle (assumed identical) and $\vec{v}_i$ the velocity. In

4) The limiting expression has an error smaller than 10^{-6} for $z > 3$.

general, $\vec{r}_i = (x_i, y_i, z_i)$ and $\vec{p}_i = (p_i^x, p_i^y, p_i^z)$ are three-dimensional vectors, although other spatial dimensions can be considered in specific cases. We can think of the particles as perfect spheres (or rods in one dimension and disks in two dimensions) moving around while interacting with other particles.

The Hamiltonian consists of two terms, corresponding to the kinetic, $\mathcal{T}$, and potential, $\mathcal{V}$, energy. In the potential energy, we must consider interactions between all possible pairs of particles (i, j), which, following convention, we order using $i < j$. As the particles have no internal degrees of freedom, the potential interaction $v(\vec{r}_i, \vec{r}_j)$ between particles i and j depends only on the location of the particles (we assume it does not depend on their velocities). The Hamiltonian is, then

$$\mathcal{H}(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}}; \vec{p}_1, \dots, \vec{p}_{\mathbf{N}}) = \mathcal{T}(\vec{p}_1, \dots, \vec{p}_{\mathbf{N}}) + \mathcal{V}(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}}) \tag{5.28}$$

with

$$\mathcal{T} = \sum_{i=1}^{\mathbf{N}} \frac{\vec{p}_i{}^2}{2m}, \tag{5.29}$$

$$\mathcal{V} = \sum_{i<j} v(\vec{r}_i, \vec{r}_j). \tag{5.30}$$

We see that the pdf $e^{-\beta\mathcal{H}}$ can be split naturally as

$$e^{-\beta\mathcal{H}} = \left[\prod_{i=1}^{\mathbf{N}} e^{-\beta \frac{\vec{p}_i{}^2}{2m}}\right] \times e^{-\beta \sum_{i<j} v(\vec{r}_i, \vec{r}_j)} \tag{5.31}$$

$$= \left[\prod_{i=1}^{\mathbf{N}} e^{-\frac{(p_i^x)^2}{2m/\beta}} e^{-\frac{(p_i^y)^2}{2m/\beta}} e^{-\frac{(p_i^z)^2}{2m/\beta}}\right] \times e^{-\beta \sum_{i<j} v(\vec{r}_i, \vec{r}_j)} \tag{5.32}$$

which indicates that, from the statistical point of view, each one of the momentum coordinates (p_i^x, p_i^y, p_i^z) is independently distributed with a Gaussian distribution of zero mean and variance $m/\beta = kTm$. This independence allows us to obtain analytically some averages of interest. For instance, the average value of the kinetic energy is

$$\left\langle \sum_{i=1}^{\mathbf{N}} \frac{\vec{p}_i{}^2}{2m} \right\rangle = \frac{3}{2}\mathbf{N}kT \tag{5.33}$$

as each of the $3\mathbf{N}$ Gaussian variables contributes a factor $kT/2$ (energy equipartition theorem). On the other hand, averages of functions $G(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}})$, which depend on the coordinates, are much more difficult to perform analytically because of the interaction terms. Here is where the numerical methods are useful. The formal expression is

$$\langle G(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}}) \rangle = C^{-1} \int d\vec{r}_1 \dots d\vec{r}_{\mathbf{N}} G(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}}) e^{-\beta \mathcal{V}(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}})} \tag{5.34}$$

or an average with respect to the pdf

$$f(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}}) = C^{-1} e^{-\beta \mathcal{V}(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}})} \tag{5.35}$$

with $C = \int d\vec{r}_1 \dots d\vec{r}_{\mathbf{N}} e^{-\beta \mathcal{V}(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}})}$, the normalization constant.

To perform these averages numerically, we generate configurations of positions $X \equiv (\vec{r}_1, \dots, \vec{r}_{\mathbf{N}})$ distributed according to this pdf and approximate the average (5.34) by the sample average, including an estimation of the error. In order to generate the configurations, we can use, for example, the Metropolis algorithm, in which we propose a change from $X \to X'$ according to some distribution $g(X'|X)$. We have much freedom in choosing the new configuration X', but we must do so such that the resulting acceptance probability $h(X'|X)$ is not vanishingly small. As explained earlier, we choose a proposal X' that differs from X in just a few variables. For instance, we can choose, most naturally, to change the position $\vec{r}_i \to \vec{r}\,'_i$ of a single particle, randomly chosen among the $\mathbf{N}$ particles. In this way, and similar to (4.71), $g(X'|X)$ is constructed from $g(\vec{r}\,'_i|\vec{r}_i)$, which depends only on the coordinates of the randomly chosen particle i. We can choose, for instance (but we stress that we have a lot of freedom in this proposal step), to change every Cartesian coordinate $(x_i, y_i, z_i) \to (x'_i, y'_i, z'_i)$ such that x'_i, y'_i, and z'_i are drawn from a uniform distribution in the intervals $(x_i - \Delta, x_i + \Delta)$, $(y_i - \Delta, y_i + \Delta)$ and $(z_i - \Delta, z_i + \Delta)$, respectively. As this proposal is symmetrical ($g(X'|X) = g(X|X')$), the detailed balance condition is

$$h(X'|X)e^{-\beta\mathcal{V}(X)} = h(X|X')e^{-\beta\mathcal{V}(X')} \tag{5.36}$$

in terms of the potential energy $\mathcal{V}$ only. We can take, for instance, the Metropolis solution

$$h(X'|X) = \min\left[1, e^{-\beta\Delta\mathcal{V}}\right] \tag{5.37}$$

where $\Delta\mathcal{V} = \mathcal{V}(X') - \mathcal{V}(X)$ is the change in the potential energy induced by the proposed change $X \to X'$. As only the position $\vec{r}_i$ is modified, the change is

$$\Delta\mathcal{V} = \sum_{j\neq i}\left[v(\vec{r}\,'_i, \vec{r}_j) - v(\vec{r}_i, \vec{r}_j)\right]. \tag{5.38}$$

Usually, the potential interaction depends on the separation distance $r_{ij} = |\vec{r}_i - \vec{r}_j|$ between particles i and j, $v(\vec{r}_i, \vec{r}_j) = v(r_{ij})$. This is the case, for example, of the celebrated Lennard–Jones potential

$$v(r) = v_0\left[\left(\frac{r_0}{r}\right)^{12} - \left(\frac{r_0}{r}\right)^{6}\right], \tag{5.39}$$

where v_0 and r_0 are the parameters of the potential.

Note that in this case, and for a large number of particles $\mathbf{N}$, the calculation of the sum (5.38) can be very expensive from the computational point of view because a sum of $\mathbf{N}$ terms must be calculated every time a single position is proposed to change. This is another reason why collective update proposals, in which all variables $(\vec{r}_1, \dots, \vec{r}_{\mathbf{N}})$ are proposed to change simultaneously, are more effective. A particular type of collective updating suitable for this kind of systems and known as the *hybrid Monte Carlo* will be explained in Chapter 10.

In some cases, the potential interaction $v(\vec{r}_i, \vec{r}_j)$ is such that it vanishes whenever the separation distance $|\vec{r}_i - \vec{r}_j|$ is larger than a cutoff R. This is typically

implemented in the Lennard–Jones potential as

$$v(r) = \begin{cases} v_0 \left[\left(\frac{r_0}{r} \right)^{12} - \left(\frac{r_0}{r} \right)^{6} \right], & r \le R, \\ 0, & r > R. \end{cases} \tag{5.40}$$

In this case, the sum (5.38) has a much smaller number of terms:

$$\Delta \mathcal{V} = \sum_{j, |\vec{r}\,'_i - \vec{r}_j| < R, |\vec{r}_i - \vec{r}_j| < R} \left[v(\vec{r}\,'_i, \vec{r}_j) - v(\vec{r}_i, \vec{r}_j) \right]. \tag{5.41}$$

However, some nontrivial bookkeeping is necessary in order to keep track of which particles j are at a distance less than the cutoff from the particle i whose position is proposed to change to $\vec{r}\,'_i$.

The main features of the Lennard–Jones potential are the presence of a repulsion term (r^{-12}) at short distances and an attraction term (r^{-6}) decreasing to zero at long distances. These features can be captured by simpler models. For instance, the potential

$$v(r) = \begin{cases} \infty, & r \le 2\sigma, \\ -v_0, & 2\sigma < r \le R, \\ 0, & r > R \end{cases} \tag{5.42}$$

signifies that the attraction between particles occurs only if they are closer than a distance R and that the repulsion energy is infinite if particles try to get closer than a distance 2σ, modeling the so-called *hard-core* repulsion. This can be imagined as each particle being a sphere of radius σ, such that the energetic cost of bringing two particles closer than a distance of one diameter, 2σ, is infinite. When only the repulsion term is taken into consideration, that is, when the potential is

$$v(r) = \begin{cases} \infty, & r \le 2\sigma, \\ 0, & r > 2\sigma \end{cases} \tag{5.43}$$

we talk about a system of hard spheres. In this particular case, the energy change $\Delta \mathcal{V}$ is either zero, if the new position $\vec{r}\,'_i$ is such that the particle does not overlap with any other, or infinite, in case there is one particle j with which it overlaps. The acceptance probability is, hence

$$h(X'|X) = \min \left[1, e^{-\beta \Delta \mathcal{V}} \right] = \begin{cases} 0 & \text{if exists } j \text{ such that } |\vec{r}\,'_i - \vec{r}_j| < 2\sigma, \\ 1, & \text{otherwise.} \end{cases} \tag{5.44}$$

In other words, movements are accepted if and only if they do not lead to overlaps between particles. Again, the program can be made much more efficient by making an *a priori* list containing the particles with which particle i can overlap after it is moved. As for a change of Δ in each coordinate the maximum change in the modulus of the position is $\sqrt{3}\Delta$, the list must include all particles that are at a distance shorter than $2\sigma + \sqrt{3}\Delta$ from particle i. Simple as it might look, the system of hard spheres has been extensively studied both from the theoretical and

numerical points of view. Very efficient codes have been developed to this end, and we refer the interested reader to the more specialized literature in this subject.

5.4 Ising Model

We have already introduced the Ising model. Although it has been used in very different contexts (from phase separation in binary metal alloys to segregation in urban communities), its most direct application (and the one it was originally introduced for) is that of magnetic materials. Imagine a substance displaying a paramagnetic to ferromagnetic transition. This is to say, the magnet loses its spontaneous magnetization at high temperatures (above the so-called Curie temperature) [5]. We can simplify the complicated structure of the magnetic solid by a perfect, regular lattice and assume that in every site i of this lattice lies a microscopic magnet that is capable of taking two possible values $x_i = \mu s_i = \pm\mu$ for the magnetic moment. Here, s_i is called the *spin* variable (a name reflecting the fact that the magnetism has its origin in the individual atomic spins). The value $s_i = +1$ means that the magnet points upward in an arbitrary, Z, direction, while a value $s_i = -\mu$ indicates that the magnet points downward in the opposite direction. A microscopic configuration $S = (s_1, \ldots, s_N)$ is a set of values for the N spin variables.

The last ingredient is an energy function, which is a Hamiltonian[6] reflecting the magnetic interaction between the spins. In ferromagnetic materials, the interaction is such that it favors two spins to point in the same direction. This is reflected in a potential interaction between the spins at sites i and j equal to $-J_{ij}s_is_j$, with J_{ij}, the *coupling constant* between sites i and j, a positive parameter. Hence, if s_i and s_j are both parallel, that is, both take the value $+1$ or the value -1, then the interaction energy is $-J_{ij}$, whereas if they are antiparallel, one of them $+1$ and the other -1, the interaction energy is $+J_{ij}$, which is higher (for $J_{ij} > 0$) than in the case of parallel alignments. The interaction energy J_{ij} decays with the distance between the sites i and j. An important simplification of the model is to assume that the interaction is very short-ranged and occurs only for those spins that are sufficiently close in the regular lattice[7]. What is meant by ''sufficiently close'' depends on the type of lattice, but usually one adopts the point of view that only those spins that lie separated by the minimum distance dictated by the lattice are capable of interacting between them. Those spins are then said to be the ''nearest neighbors'' in the lattice. In Figure 5.1, we plot some common lattices and the underlying structure of the nearest neighbors.

5) We might not be very familiar with this situation, as the most common magnetic material, iron, loses its magnetization at around 770°C, certainly not a temperature we come across every day.
6) We stress again that there are no Hamilton's equations associated with this function.
7) Another simplification of the model goes in the opposite direction, and assumes a fully connected lattice in which all spins interact with all others with the same energy. This is the so-called mean-field version of the Ising model. It is less realistic, but its main advantage is that it can be solved analytically.

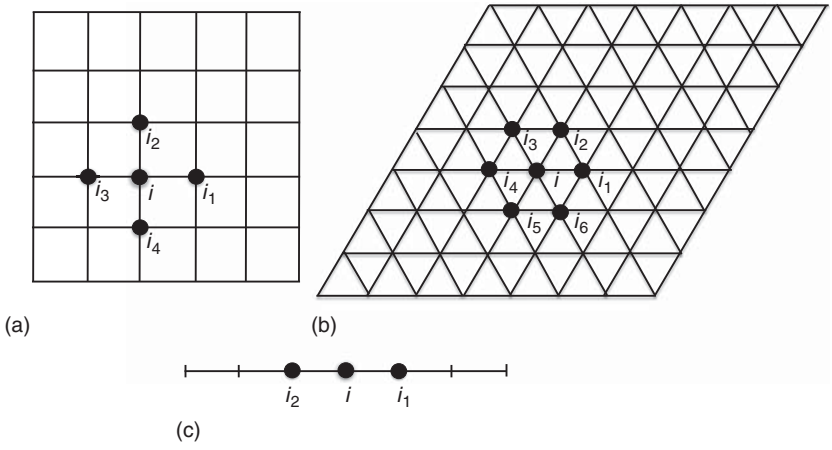


Figure 5.1 Some common lattices and the structure of their nearest neighbors. (a) In a square lattice, node i has four nearest neighbors: i_1, i_2, i_3, and i_4. (b) In a triangular lattice, node i has six nearest neighbors. i_1, i_2, i_3, i_4, i_5, and i_6. (c) In a linear chain, node i has two nearest neighbors: i_1, and i_2.

The Hamiltonian, finally, considers the possible existence of a magnetic field H with which spins tend to align. This is reflected by a term $-Hs_i$ which takes its minimum value whenever the signs of H and s_i coincide. With all these considerations in mind, the Hamiltonian of the Ising model is

$$\mathcal{H}(s_1, \ldots, s_N) = -J \sum_{\langle i,j \rangle} s_i s_j - H \sum_i s_i \tag{5.45}$$

where the notation $\langle i,j \rangle$ indicates precisely all the pairs of sites i, j that are nearest neighbors in the chosen network.

In the absence of a magnetic field, $H = 0$, the basic phenomenology of the Ising model is that the tendency to align parallel dominates at low temperatures and there is a vast majority of spins pointing in the same direction, a situation identified with macroscopic order. Whether this direction is up, $s_i = +1$, or down, $s_i = -1$, depends on many things (e.g., the initial conditions). The choice of one of the two, which are otherwise equivalent, directions is an example of "symmetry breaking." At temperatures above a critical value, T_c, the disordering role dominates and approximately half of the spins point upward and half downward. The symmetry has been restored. This competition between an ordering agent (the coupling constant) and a disordering one (the temperature) and the resulting transition between order and disorder is arguably the simplest example of a "phase transition," the phases being the ordered state at low temperatures (the ferromagnetic phase) and the disordered state at high temperatures (the paramagnetic phase). This competition between ordering and disordering agents and the resulting phase transition phenomenon can be found in many situations, not only in physical systems, and the Ising model is then used as a paradigmatic case study. An example far from the topic of magnetic materials occurs in the field of opinion

formation, where positive and negative opinions over a given topic can coexist with approximately half of the population supporting one or the other, or one of them dominating. Whatever the interpretation, it is usual to keep the magnetic notation and talk about a ''paramagnetic'' and a ''ferromagnetic'' phase, or ''spin interaction,'' even though the interpretation of the model is far from the original domain of magnetism for which it was designed.

Which averages of functions G of the spin variables $(s_1, \dots, s_N)$ are useful to compute in the Ising model? Without doubt, the most important one is the magnetization per particle, defined as

$$\mathrm{m} = \left| \frac{1}{N} \sum_{i=1}^{N} s_i \right| \tag{5.46}$$

and its ensemble average m, given by

$$m = \langle \mathrm{m} \rangle . \tag{5.47}$$

Averages of any function $G(S)$ are performed using the Boltzmann factor $\mathrm{e}^{-\beta \mathcal{H}}$, or

$$\langle G \rangle = \frac{\sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} G(s_1, \dots, s_N) e^{-\beta \mathcal{H}}}{\sum_{s_1=\pm 1} \cdots \sum_{s_N=\pm 1} e^{-\beta \mathcal{H}}}. \tag{5.48}$$

Other important quantities are the magnetic susceptibility $\chi_T = \left. \frac{\partial m(T,H)}{\partial H} \right|_{H=0}$, which can be related to fluctuations of the order parameter:

$$\chi_T = \frac{N}{kT} \sigma^2[\mathrm{m}] = \frac{N}{kT} \left[\langle \mathrm{m}^2 \rangle - \langle \mathrm{m} \rangle^2 \right], \tag{5.49}$$

the internal energy per particle, $u = \mathcal{U}/N$, given by

$$u = \frac{\langle \mathcal{H} \rangle}{N}, \tag{5.50}$$

and the specific heat per particle, $c = C/N$, which can be related to the fluctuations of energy

$$c = \frac{\sigma^2[\mathcal{H}]}{kT^2 N} = \frac{1}{kT^2 N} \left[\langle \mathcal{H}^2 \rangle - \langle \mathcal{H} \rangle^2 \right], \tag{5.51}$$

and many others. Relations of this kind between a response function, χ_T, c, and microscopic fluctuations, $\sigma^2[\mathrm{m}]$, $\sigma^2[\mathcal{H}]$, were first obtained by Einstein and go under the general name of fluctuation–dissipation relations.

The magnetization measures the degree of order: if all spins point in the same direction (either +1 or −1), then it takes the maximum value $m = 1$. If the spins point randomly in both directions, then the sum $\sum_{i=1}^{N} s_i$ is close to 0 and, after dividing by N, we get that the magnetization goes to zero as the system size N increases. It is clear from its definition that the magnetization depends on both the temperature T and the magnetic field H, $m(T, H)$. The value at zero magnetic field, $m(T, 0) \equiv m_0(T)$, is called the *spontaneous magnetization*. Again, the notation is the one used in magnetic systems: a piece of iron in a magnetic field will display some magnetization and, if the magnetic field is turned off, the magnetization

will remain different from zero only below the critical temperature. The detailed calculation for the Ising model shows that this behavior is reproduced depending on the type of lattice. For one-dimensional lattices in which spins have only two nearest neighbors, which is the case considered by Ising himself, the spontaneous magnetization is always zero, independent of temperature. However, the generic behavior for lattices in two or more dimensions is the existence of a critical temperature below which the spontaneous magnetization is indeed different from zero. From the analytical point of view, only a limited set of lattices can be studied, including a variety of two-dimensional lattices and the fully connected lattice in which every spin is connected to every other spin. It is worth mentioning here the work by Onsager who in an authentic mathematical tour de force was able to compute the free energy and the related thermodynamic potential in the case of zero magnetic field for the regular square lattice. He was also able to find the spontaneous magnetization. Despite the tremendous difficulty of the calculation, the spontaneous magnetization is given by a deceptively simple expression

$$m_0(T) = \begin{cases} 0, & T > T_c \\ \left(1 - [\sinh(2J/kT)]^{-4}\right)^{1/8}, & T \leq T_c \end{cases} \tag{5.52}$$

with a value of the critical temperature $kT_c/J = 2/\log(1+\sqrt{2}) \approx 2.2691853\ldots$.

The alternative to the complicated analytical calculations is the use of the Monte Carlo method. There, we replace the true average by the sample average

$$m = \frac{1}{M}\sum_{k=1}^{M} \mathfrak{m}_k \tag{5.53}$$

where $\mathfrak{m}_k$ is the value of the magnetization computed in $k = 1, \ldots, M$ spin configurations $S^1, \ldots, S^M$. Remember that each configuration S is a set of values for the N spin variables $S = (s_1, \ldots, s_N)$. The configurations S^k must be generated according to the probability

$$f(S) = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}(S)}. \tag{5.54}$$

5.4.1 Metropolis Algorithm

Let us consider first the Metropolis algorithm to generate the configurations S^k. The basic ingredient is the proposal probability $g(S'|S)$. We should not be surprised now of the general strategy: the proposed configuration S' differs from S only in the value of a single spin variable. Therefore, we select a site, say i, and propose a change $s_i \to s_i'$. It should be clear that the only possible proposal is $s_i' = -s_i$, that is, to propose $s_i' = +1$ if $s_i = -1$ and $s_i' = -1$ if $s_i = +1$. What is the change in the Hamiltonian $\Delta\mathcal{H} = \mathcal{H}(S') - \mathcal{H}(S)$ associated with this proposal? As $N-1$ spins remain unchanged, the only variation in the Hamiltonian comes from the terms of the sum in (5.45) in which the selected spin s_i appears. Let us denote by $s_{i_1}, \ldots, s_{i_D}$

the set of neighbors of s_i. Therefore, the change is

$$\Delta\mathcal{H} = \left(-J\sum_{\mu=1}^{D} s_i' s_{i_\mu} - Hs_i'\right) - \left(-J\sum_{\mu=1}^{D} s_i s_{i_\mu} - Hs_i\right) \tag{5.55}$$

or using $s_i' = -s_i$

$$\Delta\mathcal{H} = \left(2J\sum_{\mu=1}^{D} s_{i_\mu} + 2H\right) s_i. \tag{5.56}$$

The acceptance probability can be chosen as $h(S'|S) = \min\left[1, e^{-\beta\Delta\mathcal{H}}\right]$, the Metropolis choice, or $h(S'|S) = \left(1 + e^{\beta\Delta\mathcal{H}}\right)^{-1}$, the Glauber choice, or any other convenient expression.

Let us now give some details of the programming in the case of a regular 2-D square lattice. Most of what we will say can be straightforwardly extended to other lattices, regular or random. The first thing we need to do is to store the configuration $(s_1, \ldots, s_N)$. As the variables are ± 1, it is possible to use very sophisticated storage methods where each variable occupies only one bit of memory (a bit equal to 0 corresponds to $s_i = -1$ and a bit equal to 1 to $s_i = +1$). However, we start by a simple storage method in which spin s_i is stored in location $s(i)$ of an array of integers. We then define `integer s(N)` as the array where we store the variables. As we are in a square lattice, the number of sites is $N = L^2$, and we could as well store the variables in a two-index array, such as `integer s(L,L)`. Both notations are equivalent (Figure 5.2). However, the use of a single index brings many simplifications to the program structure and we will keep this way of storing the variables. It is convenient to keep in mind that if we use Cartesian coordinates (ix, iy), $ix, iy = 1, \ldots, L$, the single index $i = 1, \ldots, N$ is given by $i = (iy - 1) \times L + ix$, as the reader can check.

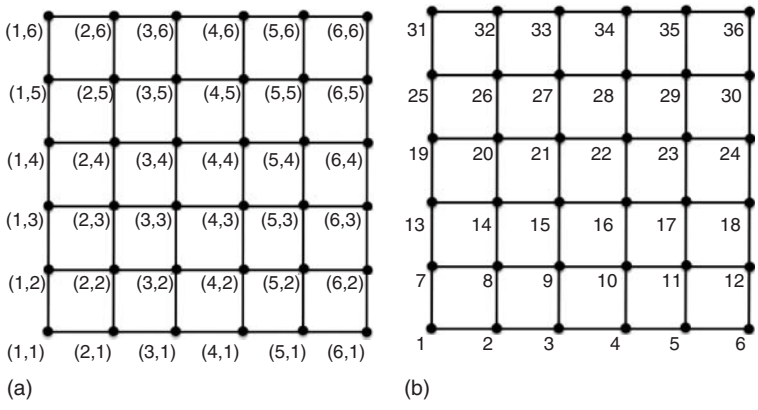


Figure 5.2 Storage of the nodes of a square lattice using (a) two indexes (i_x, i_y) (b) or a single index i. The figure exemplifies these two possibilities in the case of a linear side $L = 6$.

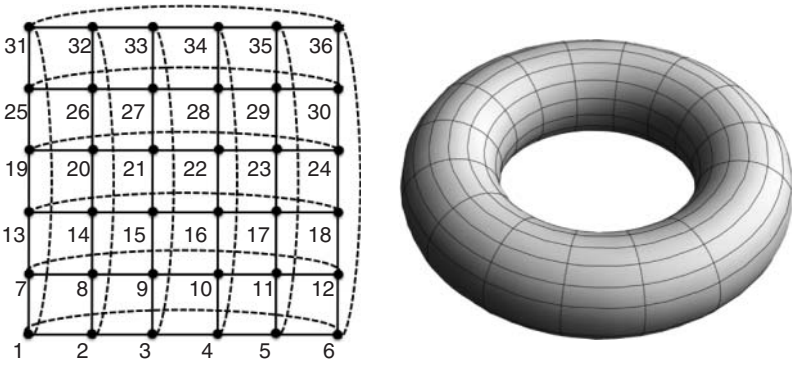


Figure 5.3 Nearest neighbor connectivity in a square lattice using periodic boundary conditions and the resulting torus topology. For example, the nearest neighbors of site $i = 12$ are $i_1 = 7$, $i_2 = 18$, $i_3 = 11$, and $i_4 = 6$, while the nearest neighbors of site $i = 31$ are $i_1 = 32$, $i_2 = 1$, $i_3 = 36$, and $i_4 = 25$.

In the square lattice, spin s_i has $D = 4$ neighbors which we name $s_{i_1}, s_{i_2}, s_{i_3}, s_{i_4}$. These neighbors are located, respectively, at the right, top, left, and bottom sites of i (Figure 5.1). It is convenient, instead of computing every time the indexes i_1, i_2, i_3, i_4 that correspond to site i, to store those values on arrays, such that they can be easily (and quickly) accessed when needed. We use the array `integer n1(N)` such that the value `n1(i)` is equal to i_1, the right neighbor of site i. Similarly, the arrays `integer n2(N), n3(N), n4(N)` store the locations of the up (i_2), left (i_3), and down (i_4) neighbors of site i. A case of special interest is the spins at the borders of the square lattice. One might simply consider that the spins at these borders do not have as many neighbors as the others. A more commonly used choice is that of periodic boundary conditions, or pbcs for short. This means that the right neighbor of a site which is located at the right edge of the square is located in the same row at the left edge: and similarly for other directions. Figure 5.3 sketches the nearest neighbor connectivity in the square lattice. It is as if the right and left edges, and the top and the bottom ones, are connected. The reader might try to imagine how this would look like. The resulting closed structure has the topology of a torus.

In order to generate the right values for those arrays, we use the `subroutine neighbors`. It uses conveniently the equivalence between the one-index i and the two-index (i_x, i_y) notations to find the four neighbors of a site. This routine implements the pbcs. For example, the right neighbor of site $(i_x = L, i_y)$ is not $(L + 1, i_y)$ but $(1, i_y)$, and so on. We urge the reader to have a look at this routine now to understand how it works.

```
subroutine neighbors(n1,n2,n3,n4,L)
dimension n1(L*L),n2(L*L),n3(L*L),n4(L*L)
  do ix=1,L
  do iy=1,L
    i=(iy-1)*L+ix
    ix1=ix+1
```

```
      if (ix1.eq.L+1) ix1=1
      n1(i)=(iy-1)*L+ix1
    iy2=iy+1
      if (iy2.eq.L+1) iy2=1
      n2(i)=(iy2-1)*L+ix
    ix3=ix-1
      if (ix3.eq.0) ix3=L
      n3(i)=(iy-1)*L+ix3
    iy4=iy-1
      if (iy4.eq.0) iy4=L
      n4(i)=(iy4-1)*L+ix
  enddo
  enddo
end subroutine neighbors
```

Next thing to consider is the calculation of the acceptance factor. We note that the possible change of energy, as given by (5.56), in the case a magnetic field $H = 0$ can take only five possible values, namely

$$-\beta\Delta\mathcal{H} = -2\beta J s_i \sum_{\mu=1}^{4} s_{i_\mu} = -2KB_i = -2K \begin{cases} -4 \\ -2 \\ 0 \\ 2 \\ 4 \end{cases} \tag{5.57}$$

where we have defined $K = \beta J$ and $B_i = s_i \sum_{\mu=1}^{4} s_{i_\mu}$, being a sum of four numbers, each one being ± 1, can only take the values $-4, -2, 0, 2, 4$, as indicated. Similarly, the Boltzmann–Gibbs factor

$$e^{-\beta\Delta\mathcal{H}} = e^{-2KB_i} \tag{5.58}$$

can only take five possible values. We decide to store the acceptance probabilities in the array `h(-4:4)`. This is defined as

$$\texttt{h(j)} = \min[1, e^{-2Kj}], \qquad j = -4, -2, 0, 2, 4. \tag{5.59}$$

To accept, we compare `h(j)` with a random number u in the usual way. If $u <$`h(j)`, we accept the proposal. Otherwise, it is discarded and we must make a new proposal by selecting randomly another spin variable.

All the preliminary steps are now defined. The program, after creating the arrays of neighbors and setting the initial condition randomly assigning values $s_i = +1$ and $s_i = -1$ with probability 1/2, then performs the proposal/acceptance steps. These are divided into two blocks: first, we perform `M0*N` thermalization steps, which is equivalent to `M0` Monte Carlo steps (MCSs) (remember that one MCS is equal to N basic proposal/acceptance steps); next, we begin the `M` measurements. Before each measurement, we perform `mc*N` updates, or `mc` MCS. In this simple implementation, we measure three things: the magnetization $\langle$`m`$\rangle$ (stored in `rm`), its fluctuations $\langle$`m`$^2\rangle - \langle$`m`$\rangle^2$ (stored in `rm2`), and the correlation function $\rho_m(t =$ `mc` MCS), or the correlation between two consecutive values of the magnetization (stored in `c`). Finally, the program runs for different values of temperature. In a setup like this, it is convenient to start first at a high

value of the temperature ($T = 4$ in the listing below), where the thermalization times are smaller. When lowering the temperature, we do not generate again the initial condition, but use instead the final configuration at the previous, higher, temperature. We now provide a full listing and ask the reader to go through all steps carefully.

```
program Ising_2D_Metropolis
  parameter (L=80,N=L*L)
  implicit double precision(a-h,o-z)
  integer s(N),n1(N),n2(N),n3(N),n4(N)
  dimension h(-4:4)
  data M,M0,mc /8192,1000,1/
  call neighbors(n1,n2,n3,n4,L)
  do i=1,N                              !Initial condition
    if (ran_u() < 0.5d0) then
      s(i)=+1
    else
      s(i)=-1
    endif
  enddo
  do 999 T=4.0,0.1,-0.1          ! Loop over temperatures
  do j=-4,4,2                    ! Create the array with the
      h(j)=min(1.0,exp(-2*j/T)) ! Boltzmann-Gibbs factors
  enddo
  do  ij=1,M0*N                  ! Thermalizing steps
      i=i_ran(N)
      ib=s(i)*(s(n1(i))+s(n2(i))+s(n3(i))+s(n4(i)))
      if (ran_u() < h(ib)) s(i)=-s(i)
  enddo
  c=0.0                          ! Initialize averages
  rm=0.0
  rm2=0.0
  rm1=real(abs(sum(s)))/N
  do  im=1,M                     ! Updating steps
    do ij=1,mc*N
      i=i_ran(N)
      ib=s(i)*(s(n1(i))+s(n2(i))+s(n3(i))+s(n4(i)))
      if (ran_u() < h(ib)) s(i)=-s(i)
    enddo
    rm0=real(abs(sum(s)))/N      !  Begin measures
    write(88,*) rm0
    rm=rm+rm0
    rm2=rm2+rm0*rm0
    c=c+rm0*rm1
    rm1=rm0
  enddo
!Final averages
  rm=rm/M
  rm2=rm2/M-rm*rm
```

```
  c=(c/M-rm*rm)/rm2
  if (c.ne.1.0) tau=c/(1.0d0-c)
  error=sqrt(rm2*(2*tau+1)/M)
  write(66,'(f10.6,1p5e16.6)') T,rm,rm2,error,mc*tau,c
999 continue
end program  Ising_2D_Metropolis
```

The kernel of the program are the three lines

```
i=i_ran(N)
ib=s(i)*(s(n1(i))+s(n2(i))+s(n3(i))+s(n4(i)))
if (ran_u() < h(ib)) s(i)=-s(i).
```

The first line selects randomly the spin to be updated. The second line computes the necessary index of the acceptance probability. The third line does the actual updating. In fact, if `h(ib)` ≥ 1, something that occurs if `ib` ≤ 0, there is no need to compare the acceptance probability with a random number, as the proposal is always accepted. This can be implemented by modifying the last line to

```
if (ib <= 0) then
 s(i)=-s(i)
else if (ran_u() < h(ib)) s(i)=-s(i)
endif
```

which reduces the number of calls to the random number generator routine but increases the complexity of the program. If we would like to implement other updating probabilities, such as Glauber, all we would need to do is to change

```
do j=-4,4,2
  h(j)=1.0d/(1.0+exp(2*j/T))
enddo
```

Another widely used modification is to run sequentially through the sites to update instead of selecting them sequentially. This can be done by choosing the spins in the order $s_1, s_2, s_3, \ldots, s_N$. This requires only a simple modification of the program. Namely, replace the lines around `i=i_ran(N)` by

```
do ij=1,mc
 do  i=1,N
    ib=s(i)*(s(n1(i))+s(n2(i))+s(n3(i))+s(n4(i)))
    if (ran_u() < h(ib)) s(i)=-s(i)
 enddo
enddo
```

Finally, note that the program stores in `unit 88` each one of the values of the magnetization. This is necessary, for example, to compute accurately the correlation time τ_m of this quantity using another program, such as the one explained in Appendix C. The program given here uses, instead, as a rough estimate for the correlation time the value $\tau_m = \frac{\rho_m(1)}{1-\rho_m(1)}$ derived in (4.26).

In Figure 5.4, we plot some representative configurations obtained after running this basic Ising model program. We can see the order–disorder phenomenology: at temperatures larger than the critical temperature T_c, the system is disordered and there is, on a local average, approximately the same number of up and down spins. At low temperatures, one of the two spin options dominates. At $T \approx T_c$, there is still the same average number of up and down spins, but they begin to organize themselves in a fractal-like self-similar structure, the precursor of the phase transition.

5.4.2
Kawasaki Interpretation of the Ising Model

We mentioned already that the Ising model can be used in very many different contexts. An interesting interpretation as a model for binary alloy separation was introduced by Kawasaki. In his model, a variable taking a value $s_i = +1$ indicates that on node i there is one atom, A, of a particular metal (e.g., aluminum), whereas $s_i = -1$ indicates the presence of another type of atom B (e.g., zinc). The Hamiltonian is still given by (5.45) but now the magnetic field H is interpreted as the difference in chemical potential between the two types of atoms. The novelty compared to the usual interpretation as a ferromagnet is that atoms A and B cannot transmute. This means that the number of A atoms and the number of B atoms are both constant. The averages have to be performed with respect to the same pdf as before, $f = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}}$, but only the total number of A atoms is a prefixed number N_A (and consequently the number of B atoms is $N_B = N - N_A$.

How do we implement such a constraint in our proposal probability $g(S'|S)$? Kawasaki's proposal is to allow for changes $S \to S'$ by which two neighboring atoms exchange positions. This certainly keeps both the number of A and B atoms constant; they simply move around. It is also required to mimic the real diffusion process that occurs in an alloy as it is cooled down from a high-temperature phase. So, in the proposal step, we select a pair of neighbor sites i, j, check that $s_i = -s_j$ (otherwise, it does not matter whether we accept the change or not), and propose the change $(s_i, s_j) \to (s'_i, s'_j)$ with $s'_i = s_j = -s_i, s'_j = s_i = -s_j$. It is clear that this is a symmetric proposal, $g(S'|S) = g(S|S')$. In order to choose the pair of neighboring sites, we first choose i randomly between 1 and N. Then, in the square lattice, it suffices to select j with equal probability as the neighbor to the right or the neighbor at the top. Imagine we have selected j as the neighbor to the right, that is, $s_j = s_{i_1}$. Then, the only terms of the Hamiltonian that will change after exchanging s_i and s_j will be $s_i(s_{i_2} + s_{i_3} + s_{i_4}) + s_j(s_{j_1} + s_{j_2} + s_{j_4})$ (it will help to understand this if the reader plots the lattice and the involved spins). The change in energy will be

$$\begin{aligned} \Delta \mathcal{H} = &-J \left(s'_i(s_{i_2} + s_{i_3} + s_{i_4}) + s'_j(s_{j_1} + s_{j_2} + s_{j_4}) \right) \\ &+J \left(s_i(s_{i_2} + s_{i_3} + s_{i_4}) + s_j(s_{j_1} + s_{j_2} + s_{j_4}) \right) \end{aligned} \tag{5.60}$$

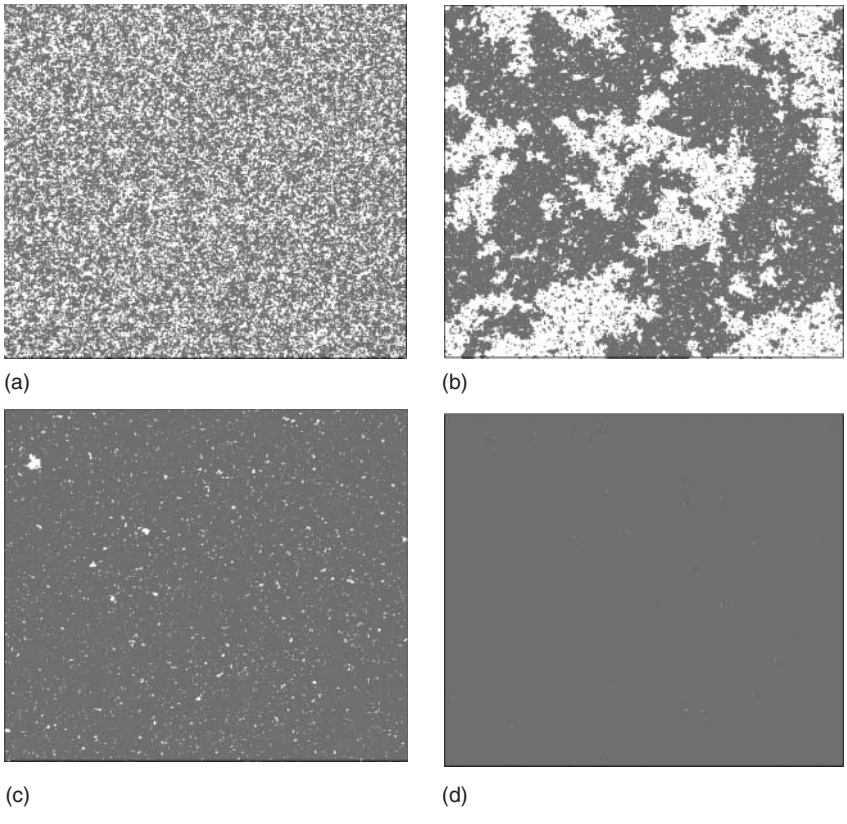


Figure 5.4 Representative configurations of the Ising model at (a) $T = 4$, (b) $T = T_c$, (c) $T = 2$, and (d) $T = 1$, for a system of $N = 400^2$ sites. A spin up, $s_i = +1$, is indicated by a dark dot, whereas a spin down, $s_i = -1$, is represented by a white dot. Observe that, at $T = T_c$, the structure is fractal, meaning that there are many blocks of up spins in a background of down spins, and vice versa, recursively.

or using $s'_j = -s'_i = -s_j = s_i$

$$\Delta\mathcal{H} = 2Js_i \left(s_{i_2} + s_{i_3} + s_{i_4} - s_{j_1} - s_{j_2} - s_{j_4} \right). \tag{5.61}$$

If the site j would have been the one located up of site i, then the charge in energy would be similar but involving a different set of sites, namely

$$\Delta\mathcal{H} = 2Js_i \left(s_{i_1} + s_{i_3} + s_{i_4} - s_{j_1} - s_{j_2} - s_{j_3} \right). \tag{5.62}$$

In any of the two events, one can write $-\beta\Delta\mathcal{H} = -2KB_i$, where B_i is a number extracted from the set $(-6, -4, -2, 0, 2, 4, 6)$. Besides the selection of the spins

to be interchanged, the main modification to the above program is to change the dimension of array h from `integer h(-4:4)` to `integer h(-6:6)` and write

```
do j=-6,6,2
  h(j)=min(1.0,exp(-2*j/T))
enddo
```

The updating part is

```
1 i=i_ran(N)
  if (ran_u() < 0.5d0) then
   j=n1(i)
   if (s(i)=s(j)) goto 1
   ib=s(i)*(s(n2(i))+s(n3(i))+s(n4(i))-s(n1(j))
      -s(n2(j))-s(n4(j)))
  else
   j=n2(i)
   if (s(i)=s(j)) goto 1
   ib=s(i)*(s(n1(i))+s(n3(i))+s(n4(i))-s(n1(j))
      -s(n2(j))-s(n3(j)))
  endif
  if (ran_u() < h(ib)) then
     s(i)=-s(i)
     s(j)=-s(j)
   endif
```

Of course, now it does not make sense to compute the average magnetization, as this is a constant (it is the number of A atoms minus the number of B atoms). A typical quantity to measure instead is the number of links that join two nearest neighbor sites in which the variables take different values, a sort of contact area between the metals of the two alloys.

One of the most important drawbacks of this algorithm is that it might occur too often that the two selected sites hold the same type of atoms, and then this trial has to be repeated. A possibility is to make a list of all possible pairs of neighboring sites that hold different values for the spin variables and then choose a pair only from this list. This possibility speeds up the program but it does require some amount of programming[8)]

Considered as a real model for a binary alloy, the Kawasaki model is able to reproduce some well-known facts known to metallurgists. First, the process leads

8) This modification constitutes the *n-fold way* proposed by Bortz, Kalos, and Lebowitz. It will be met again in a different context when we discuss the numerical integration of master equations in Chapter 9.

to phase separation (with each alloy occupying mainly a localized spatial area) only below a critical temperature, T_c. For temperatures larger than T_c, the two metals are well mixed, whereas for a temperature smaller than T_c the system splits into two phases, each one of them rich in one of the metals. The second interesting feature of the Kawasaki model is that the time evolution of the phase separation process occurring at temperatures below T_c depends strongly on the initial relative proportion of each metal. If this proportion is close to 50%, then the process proceeds by what is called *spinodal decomposition* in which filament-like structures linking all the atoms of one metal begin to form and then coarsen and grow. When one of the metals is more abundant than the other, then the metal that is in minority starts forming small droplets embedded in the majority metal. These droplets then begin to coalescence and form larger droplets, until only one big one (and maybe several very minor droplets) can be seen. This evolution process can be observed in Figure 5.5.

5.4.3 Heat-Bath Algorithm

The Ising model (we now return to the magnetic version in which the number of spins of one or the other type does not need to be conserved) offers an interesting and relatively simple application of the heat-bath method. Remember that in this method the acceptance probability is always equal to 1, but only after the proposal probability has been carefully chosen (it is not arbitrary anymore) in order to ensure that the detailed balance condition is satisfied. So, we want to propose the change from configuration $S = (s_1, \dots, s_i, \dots, s_N)$ to $S' = (s_1, \dots, s_i', \dots, s_N)$ in which the only change is in one spin located at, say, site i. According to (4.77), the proposal probability $g(S'|S) \equiv g(s_i')$ is given by the conditional probability

$$g(s_i') = f(s_i'|s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N) = \frac{f(s_1, \dots, s_{i-1}, s_i', s_{i+1}, \dots, s_N)}{f(s_1, \dots, s_{i-1}, s_{i+1}, \dots, s_N)}. \tag{5.63}$$

This might look a very complicated expression (and maybe it is so), but we must take into account that we are proposing a new value only for s_i' while all other variables remain unchanged. All we have to do, then, is to look for the dependence on s_i' of the previous expression, all other variables taking simply fixed numbers: in other words, they are constants. If we look at the numerator of 5.63 and use $f = e^{-\beta \mathcal{H}}$, we realize that the only place where s_i' appears is in the term in $\mathcal{H}$ in which it is multiplied to its neighbors, namely $\sum_\mu s_i' s_{i_\mu}$, with s_{i_μ} being the set of D neighbors of s_i ($D = 4$ neighbors in the square lattice). Therefore, we have

$$g(s_i') = C^{-1} e^{\beta J s_i' \sum_\mu s_{i_\mu}} \equiv C^{-1} e^{a_i s_i'} \tag{5.64}$$

where we have defined $a_i = \beta J \sum_\mu s_{i_\mu}$ and C is the normalization constant, hiding all the nasty dependence in the rest of the spin variables. a_i can be considered as a ''local field'', namely the field that spin i sees and contains information about the neighboring spins. The nice part is that, however complicated the expression for C is if we write it in full, we have a much simpler method to compute its actual value.

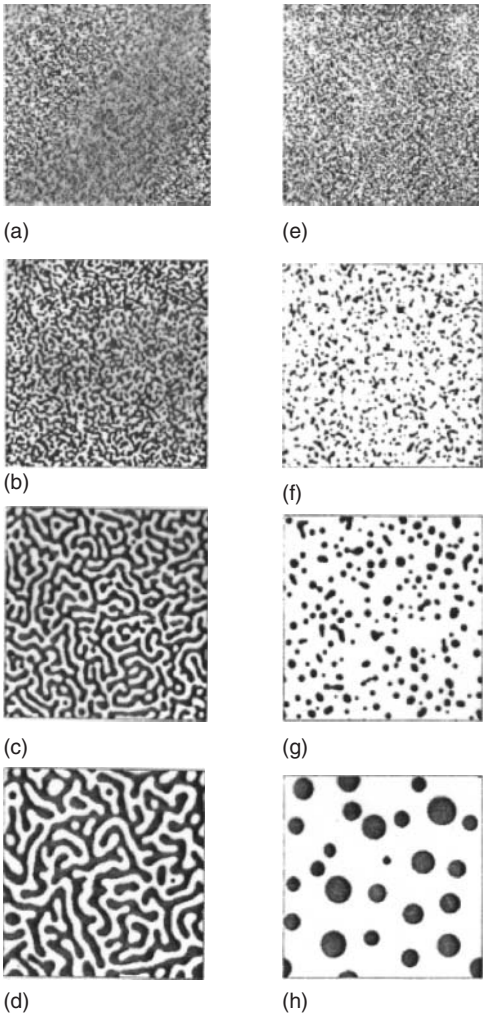


Figure 5.5 Representative configurations of the Kawasaki version of the Ising model at $T = 0.95$ starting from an initial condition with exactly half the spins in the +1 state (a–d), and starting with one-third of the spins in the +1 state (e–h). In both cases, time runs from the top to the bottom panels. In (a–d), we see the spinodal decomposition evolution mechanism, whereas in (e–h) we observe the nucleation and growth of droplets of one phase.

Simply realize that s_i' can only take the values $s_i' = \pm 1$ and hence the probabilities of these two values must add up to 1, $g(s_i' = +1) + g(s_i' = -1) = 1$, which leads immediately to $C = e^{a_i} + e^{-a_i}$, or

$$g(s_i' = +1) = \frac{e^{a_i}}{e^{a_i} + e^{-a_i}} = \frac{1}{1 + e^{-2a_i}}, \tag{5.65}$$

$$g(s_i' = -1) = \frac{e^{-a_i}}{e^{a_i} + e^{-a_i}} = \frac{e^{-2a_i}}{1 + e^{-2a_i}}. \tag{5.66}$$

The rest is simple: generate a random number u uniformly distributed in $(0, 1)$. If $u < g(s_i' = +1)$, set $s_i' = 1$, otherwise set $s_i' = -1$. This new value is always accepted. Again we can use the fact that the sum $\sum_\mu s_{i_\mu}$ can only take values from a limited set, namely $(-4, -2, 0, 2, 4)$ in the square lattice, which allows us to compute and store the possible values of $g(s_i')$. Consequently, we define `dimension g(-4:4)` and fill up the elements of this array with the corresponding values of $g(s_i' = +1)$, namely

```
do j=-4,4,2
  g(j)=1./(1.+exp(-2.*j/T)
enddo
```

and then the updating steps are

```
do ij=1,mc*N
  i=i_ran(N)
  ia=s(n1(i))+s(n2(i))+s(n3(i))+s(n4(i))
  if (ran_u() < g(ia)) then
    s(i)=+1
  else
    s(i)=-1
  endif
enddo
```

5.5 Heisenberg Model

This is also a lattice model, but the variable associated with the lattice site i is a D-dimensional vector $\vec{s}_i$ of modulus 1, $|\vec{s}_i| = 1$. The Hamiltonian is similar to that of the Ising model but it involves a scalar product

$$\mathcal{H} = -J \sum_{\langle i,j \rangle} \vec{s}_i \cdot \vec{s}_j - \vec{H} \cdot \sum_i \vec{s}_i \tag{5.67}$$

where $\vec{H}$ is the external magnetic field. It is important not to confuse the spatial dimension d of the lattice with the dimension D of the spins on the lattice. We now explain a heat-bath method in order to generate configurations $X = (\vec{s}_1, \ldots, \vec{s}_N)$ distributed according to the Boltzmann weight $f(S) = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}}$. We restrict ourselves to the $D = 3$ case, but appropriate extensions are possible to other dimensions. We do not go through all the details, which are very similar to the Ising model case. What we have to do now is to propose a value of the vector $\vec{s}_i{}'$ chosen from the distribution

$$g(\vec{s}_i{}') = C^{-1} e^{\beta (J \sum_\mu \vec{s}_{i_\mu} + \vec{H}) \cdot \vec{s}_i{}'} \equiv C^{-1} e^{\vec{a}_i \cdot \vec{s}_i{}'} \tag{5.68}$$

where C is a normalization constant and we have defined $\vec{a}_i = \beta (J \sum_\mu \vec{s}_{i_\mu} + \vec{H})$. In the heat-bath method, we need to generate a value of $\vec{s}_i{}'$ extracted from the distribution $g(\vec{s}_i{}')$ assuming that the "local field" $\vec{a}_i$ (which contains information about the other spins) is a constant.

Now, a three-dimensional vector of modulus 1 can be defined by giving the two orientation angles $\vec{s}_i{}' = (\phi_i, \theta_i)$ with $\phi_i \in (0, 2\pi)$ and $\theta_i \in (0, \pi)$. The two angles (ϕ_i, θ_i) can be taken with respect to any orientation, and it makes sense, according to the expression above, to consider them with respect to the orientation given by the local field $\vec{a}_i$. Furthermore, it is convenient to work with a new variable defined as $\xi_i = \cos(\theta_i)$ which takes values in the interval $\xi_i \in (-1, 1)$, such that the scalar product becomes $\vec{a}_i \cdot \vec{s}_i{}' = a_i \xi_i$. It is possible to find the constant C from the normalization condition

$$\int_{-1}^{+1} d\xi_i \int_0^{2\pi} d\phi_i g(\phi_i, \xi_i) = \int_{-1}^{+1} d\xi_i \int_0^{2\pi} d\phi_i C^{-1} e^{a_i \xi_i} = 1 \tag{5.69}$$

or

$$C = 2\pi \frac{e^{a_i} - e^{-a_i}}{a_i} \tag{5.70}$$

yielding a proposal pdf which can be split as $g(\phi_i, \xi_i) = g(\phi_i) \times g(\xi_i)$, with $g(\phi_i) = 1/2\pi$ a uniform distribution in the interval $(0, 2\pi)$ and

$$g(\xi_i) = \frac{a_i}{e^{a_i} - e^{-a_i}} e^{a_i \xi_i}, \quad \xi_i \in (-1, 1). \tag{5.71}$$

The angle ϕ_i is simply generated by $\phi_i = 2\pi u_i$, with u_i a $\hat{\mathbf{U}}(0, 1)$, a random number distributed uniformly in the interval $(0, 1)$. Concerning $g(\xi_i)$, it can be implemented using the general inversion method, or solving

$$v_i = \int_{-1}^{\xi_i} d\xi_i' g(\xi_i') \tag{5.72}$$

where v_i is an independent $\hat{\mathbf{U}}(0, 1)$ number. Solving this equation, we obtain

$$\xi_i = \frac{1}{a_i} \log\left(1 + (e^{2a_i} - 1)v_i\right) - 1. \tag{5.73}$$

Once ϕ_i and ξ_i have been obtained, it is a matter of algebra to find the expression of $\vec{s}_i{}'$ relative to whatever fixed coordinate system. Typically, the external field $\vec{H}$ serves to define the Z direction.

5.6 Lattice Φ^4 Model

As a final example of the application of the Monte Carlo methods in statistical physics, we consider the so-called lattice Φ^4 model. The model is the lattice version of a field model. By a field model, we mean that in every point of space $\vec{r}$ there is defined a real variable, $\Phi(\vec{r})$. In the lattice version, there are only variables Φ_i defined in the sites $i = 1, \ldots, N$ of a lattice. At variance with the Ising or Heisenberg model, where some restriction were applied to the site variable (either a binary variable or a vector of modulus 1), in this model each variable can take any real value.

In the field version, the Hamiltonian is a function of the field, namely

$$\mathcal{H}(\{\Phi(\vec{r})\}) = \int d\vec{r}\left[\frac{-B}{2}\Phi(\vec{r})^2 + \frac{U}{4}\Phi(\vec{r})^4 + \frac{K}{2}|\vec{\nabla}\Phi(\vec{r})|^2 - H(\vec{r})\Phi(\vec{r})\right] \quad (5.74)$$

where B, U, and K are the parameters of the model and $H(\vec{r})$ is an external field, usually assumed to be constant, $H(\vec{r}) = H$. This field model has been used in many different contexts and in such different topics as quantum field theory or in the study of phase transitions which is of interest here. One can think, for instance, of a fluid, and then the field $\Phi(\vec{r})$ is the density field. Of course, because of the atomic nature of matter, it is difficult to define a density field at every point of space, as this would be a highly discontinuous function. One can then parcel the space in cells forming a lattice and define in each lattice site i the variable Φ_i as the average density in the cell associated to the site. The continuous field model interpretation is preferred in the analytical calculations despite some problems that appear at very small spatial scales (reflecting precisely the discontinuity of matter at the atomic scale). It has been extensively used in renormalization group studies in momentum space. The lattice version is the preferred one for numerical calculations and provides also a regularization of the Hamiltonian of the continuous model.

For the lattice version, we consider a d-dimensional (hypercubic) regular lattice Λ consisting of $N = L^d$ points. Every point $i = 1, \dots, N$ of this lattice has d coordinates: $i = (j_1, \dots, j_d)$. The set of $2d$ neighbors have coordinates

$$\begin{aligned}
&i_1 = (j_1 + 1, \dots, j_d),\\
&\dots\dots\dots\dots\dots\dots\dots\dots\\
&i_d = (j_1, \dots, j_d + 1),\\
&i_{d+1} = (j_1 - 1, \dots, j_d),\\
&\dots\dots\dots\dots\dots\dots\dots\dots\\
&i_{2d} = (j_1, \dots, j_d - 1).
\end{aligned}$$

Periodic boundary conditions are assumed on this lattice. At every site of the lattice, there is a scalar variable Φ_i. The set of all variables is $[\Phi] \equiv (\Phi_1, \dots, \Phi_N)$. We also introduce a Hamiltonian function $\mathcal{H}$ given by

$$\mathcal{H}([\Phi]) = \sum_{i=1}^{N} a_0^d\left[\frac{-B}{2}\Phi_i^2 + \frac{U}{4}\Phi_i^4 + \frac{K}{2}\sum_{\mu=1}^{d}\left(\frac{\Phi_{i_\mu} - \Phi_i}{a_0}\right)^2 - H\Phi_i\right] \quad (5.75)$$

where the parameter a_0 (the lattice spacing) has been introduced to stress the fact that the continuous field version can be recovered (if needed) by taking the limit $a_0 \to 0$ and $L \to \infty$. Note the replacement in (5.75) of the integral of (5.74) by a sum and the spatial derivative of the gradient by a difference. If we are not interested in the continuous version, we can simply set $a_0 = 1$, as it only implies a rescaling of parameters B, U, K, and H. In applications to studies of phase transitions, the parameter B depends on temperature, $B(T)$. The first two and the last terms of the sum appearing in the Hamiltonian are local terms (depending only on the field at location i) and can be thought of as local potential terms $V(\Phi_i)$:

$$V(\Phi_i) = \frac{-B}{2}\Phi_i^2 + \frac{U}{4}\Phi_i^4 - H\Phi_i. \quad (5.76)$$

The third term in the Hamiltonian (5.75), the one multiplied by K, is called the *interaction term*, as it contains cross terms between fields in nearest neighbor sites in the regular lattice. Expanding the square and noticing that

$$\sum_{i=1}^{N}\sum_{\mu=1}^{d}\Phi_{i_\mu}^2 = d\sum_{i=1}^{N}\Phi_i^2 \tag{5.77}$$

it is possible to rewrite the Hamiltonian (after setting $a_0 = 1$) as

$$\mathcal{H}([\Phi]) = \sum_{i=1}^{N}\left[\frac{-B+2dK}{2}\Phi_i^2 + \frac{U}{4}\Phi_i^4 - K\sum_{\mu=1}^{d}\Phi_{i_\mu}\Phi_i - H\Phi_i\right]. \tag{5.78}$$

Note that the interaction terms contain all the products of nearest neighbor variables, leading to a form that is not so different from that of the Ising model:

$$\mathcal{H}([\Phi]) = \sum_{i=1}^{N}\left[\frac{-B+2dK}{2}\Phi_i^2 + \frac{U}{4}\Phi_i^4 - H\Phi_i\right] - K\sum_{\langle i,j\rangle}\Phi_i\Phi_j. \tag{5.79}$$

From a statistical mechanics analysis of this model, the importance of the Hamiltonian lies in that it dictates the pdf of the set of variables $\mathbf{\Phi} \equiv (\Phi_1, \ldots, \Phi_N)$ as

$$f(\mathbf{\Phi}) = \mathcal{Z}^{-1}e^{-\beta\mathcal{H}(\mathbf{\Phi})}, \tag{5.80}$$

where $\mathcal{Z}$ is the partition function, computed as the multiple integral of the Boltzmann factor for all the field values:

$$\mathcal{Z} = \int_{-\infty}^{\infty} d\Phi_1 \ldots \int_{-\infty}^{\infty} d\Phi_N e^{-\beta\mathcal{H}([\Phi])} \equiv \int d\mathbf{\Phi} e^{-\beta\mathcal{H}([\Phi])}. \tag{5.81}$$

For this integral to exist, that is, for the Φ^4 model to be consistent, it is required that the parameter U satisfies $U > 0$.

As usual, the magnitudes of interest are computed as averages of field functions, $G(\mathbf{\Phi})$, with the probability density function $f(\mathbf{\Phi})$:

$$\langle G(\mathbf{\Phi})\rangle = \mathcal{Z}^{-1}\int d\mathbf{\Phi} G(\mathbf{\Phi}) e^{-\beta\mathcal{H}(\mathbf{\Phi})}. \tag{5.82}$$

Example of quantities of interest are the *magnetization* m defined as

$$m = \left\langle \left| \frac{1}{N}\sum_{i=1}^{N}\Phi_i \right| \right\rangle, \tag{5.83}$$

and, particularly, its value at $H = 0$, the spontaneous magnetization m_0. Other quantities of interest are the internal energy, $\mathcal{U}$, given by

$$\mathcal{U} = \langle \mathcal{H}(\mathbf{\Phi})\rangle \tag{5.84}$$

as well as the magnetic susceptibility, χ_T, and the specific heat, defined as the fluctuations of the magnetization and the internal energy, using the same definitions (5.49)–(5.51) as in the Ising model.

Before explaining the specificity of the numerical methods of the Monte Carlo type to compute the above averages, we shall briefly review some qualitative aspects

as given by a simple approximate theory, known as the *mean-field approximation* and first introduced by Landau. The basic idea is to assume that the different variables Φ_i adopt a common value Φ_0, the mean field. The common value Φ_0 is chosen, logically, as the one that maximizes the pdf (5.80) or, equivalently, as the one that minimizes the Hamiltonian $\mathcal{H}$. After setting $\Phi_i = \Phi_0$, $\forall i$ in (5.75), the term proportional to K vanishes and all the terms in the sum are identical, leading to $\mathcal{H} = NV(\Phi_0)$. Hence, the minima of $\mathcal{H}$ are those of the local potential $V(\Phi_0)$. Let us consider first the case of no external field, $H = 0$. In this case, the local potential changes qualitatively when the parameter B changes sign. If $B < 0$, the local potential has only one minimum at $\Phi_0 = 0$. On the other hand, when $B > 0$, there are two minima of equal depth located at $\Phi_0 = \pm\sqrt{B/U}$. As the magnetization is defined as the average of the absolute value of the field, it follows the prediction of the mean-field theory for the spontaneous magnetization:

$$m_0 = \begin{cases} 0, & B < 0, \\ \sqrt{\frac{B}{U}}, & B \geq 0. \end{cases} \tag{5.85}$$

As mentioned earlier, the parameter B is assumed to be a decreasing function of temperature, such as $B(T) = B_0(T_0 - T)$, for example. The condition $B(T_0) = 0$ defines the critical temperature T_0. The magnetization is then zero for temperatures above T_0 and nonzero below T_0. This is the basic phenomenology of phase transitions and allows us to interpret m as the order parameter.

The basic picture given by the mean-field approximation is qualitatively correct[9], but not the details. It is still true that the magnetization is zero below some critical temperature T_c, but this is not determined by the condition $B(T_c) = 0$ but, instead, we find $B_c = B(T_c) \neq 0$. This can be summarized as

$$m_0 = \begin{cases} 0, & B < B_c, \\ m_0(B) \neq 0, & B \geq B_c \end{cases} \tag{5.86}$$

where B_c is dependent on the other parameters of the theory, namely U and K. The function $m(B)$ can be expanded near $B \gtrsim B_c$ as $m(B) = m_0(B - B_c)^{\beta}$, with m_0 a constant and β the critical exponent. The determination of critical exponents is one of the main points of interest in the modern theory of phase transitions. It is proved that the different phase transitions can be classified in different "universality classes," each class characterized, among other things, by a common value of the critical exponents. In particular, it is known that the Ising and the Φ^4 models at the same spatial dimension belong to the same universality class.

5.6.1 Monte Carlo Methods

After this short introduction to the phenomenology of the Φ^4 model, let us now explain the implementation of Monte Carlo methods for the numerical calculation

9) To be precise, it is correct if the spatial dimension d is larger than 1.

of averages such as the magnetization $\mathtt{m}$ of the internal energy $\mathcal{U}$. It is convenient to start first with a simplification of parameters. Let us redefine

$$\phi_i = (\beta K)^{1/2}\Phi_i, \tag{5.87}$$

$$b = \frac{B}{K} - 2d, \tag{5.88}$$

$$u = \frac{U}{\beta K^2}, \tag{5.89}$$

$$h = H\left(\frac{\beta}{K}\right)^{1/2}, \tag{5.90}$$

a rescaling chosen such that

$$\beta\mathcal{H}(\phi) = \sum_{i=1}^{N}\left[\frac{-b-2d}{2}\phi_i^2 + \frac{u}{4}\phi_i^4 + \frac{1}{2}\sum_{\mu=1}^{d}\left(\phi_{i_\mu} - \phi_i\right)^2 - h\phi_i\right], \tag{5.91}$$

or, expanding the squares as we did before

$$\beta\mathcal{H}(\phi) = \sum_{i=1}^{N}\left[\frac{-b}{2}\phi_i^2 + \frac{u}{4}\phi_i^4 - h\phi_i\right] - \sum_{\langle i,j\rangle}\phi_i\phi_j. \tag{5.92}$$

The pdf for the new rescaled variables is

$$f(\phi) = Z^{-1}e^{-\beta\mathcal{H}(\phi)} \tag{5.93}$$

with Z being the normalization constant. In this reparameterization, we are left with only three parameters: b, u, and h.

In order to generate representative configurations $\phi = (\phi_1, \ldots, \phi_N)$ distributed according to this pdf, we can use a simple[10] Metropolis algorithm. Propose a new configuration in which only one variable, say ϕ_i, changes, $\phi_i \to \phi_i'$, and all other variables remain unchanged. A possible choice is to take ϕ_i' uniformly from the interval $(\phi_i - \Delta, \phi_i + \Delta)$, with Δ being a parameter to be chosen to optimize the method. The proposal ϕ_i' is accepted with a probability

$$h(\phi'|\phi) = \min(1, e^{-\beta\Delta\mathcal{H}}). \tag{5.94}$$

In computing the change in the Hamiltonian implied by the proposed change, we do not need to use the full expression (5.92), but rather notice that most of the terms disappear when subtracting the old and the new values, such that the change is simply

$$\beta\Delta\mathcal{H} = -\frac{b}{2}(\phi_i'^2 - \phi_i^2) + \frac{u}{4}(\phi_i'^4 - \phi_i^4) - (\phi_i' - \phi_i)\left(h + \sum_{\mu=1}^{2d}\phi_{i_\mu}\right). \tag{5.95}$$

Alternatively, one could devise a heat-bath method. First, we select a site i and want to propose a value ϕ_i for the variable at that site. This is done with a proposal $g(\phi_i)$, which is proportional to $e^{-\beta\mathcal{H}}$ but where all other variables ϕ_j, $j \neq i$ are considered to be constants. All we need to consider when writing down

10) We hope that, by now, the reader finds this kind of techniques ''simple''.

$g(\phi_i)$ are those terms in the Hamiltonian where ϕ_i appears explicitly. After these considerations, we have

$$g(\phi_i) = C^{-1} e^{-\frac{u}{4}\phi_i^4 + \frac{b}{2}\phi_i^2 + b_i\phi_i} \tag{5.96}$$

with $b_i = \left(h + \sum_{\mu=1}^{2d} \phi_{i_\mu}\right)$, and C is the normalization constant, given by

$$C = \int_{-\infty}^{\infty} d\phi_i \, e^{-\frac{u}{4}\phi_i^4 + \frac{b}{2}\phi_i^2 + b_i\phi_i}. \tag{5.97}$$

We can use now any method to sample the one-variable pdf $g(\phi_i)$. As the constant C might be cumbersome to obtain in some cases, it is convenient to use a rejection method to sample $g(\phi_i)$. For instance, we could split (5.96) as

$$g(\phi) \propto \frac{1}{\sigma\sqrt{2\pi}} e^{-(\phi-\mu)^2/2\sigma^2} \times e^{-\frac{u}{4}\phi^4} \tag{5.98}$$

with $\sigma^2 = -1/b$, a procedure valid only for $b < 0$ (or $B < 2dK$ in the original parameters of the model) and $\mu = b_i/b$. Then, we could use a rejection method, consisting in proposing a value of ϕ_i according to a Gaussian distribution of mean μ and variance σ^2 and then accept that proposed value with a probability $e^{-\frac{u}{4}\phi_i^4} \in (0, 1)$. This could be programmed with the following lines:

```
1   phi=sigma*ran_g()+mu
    if (ran_u() > exp(-0.25*u*phi**4) goto 1
```

Remember that in the heat-bath method, the proposals are always accepted.

Many other choices for the proposal and acceptance probabilities are, of course, possible. A characteristic feature of the Φ^4 model, also present in many other models, is that it is possible to split the Hamiltonian into local and interaction terms. We have already explained that local terms depend only on one of the variables, whereas interaction terms depend on several variables. This is precisely what we had done in (5.92), which we rewrite as

$$\mathcal{H} = \sum_{i=1}^{N} v(\phi_i) + \mathcal{H}_I \tag{5.99}$$

with

$$\beta v(\phi_i) = -\frac{b}{2}\phi_i^2 + \frac{u}{4}\phi_i^4 - h\phi_i, \tag{5.100}$$

$$\beta\mathcal{H}_I = -\sum_{\langle i,j\rangle} \phi_i\phi_j. \tag{5.101}$$

We now use an approach in which the proposal depends only on the local term while the acceptance probability will take care of the interaction terms. Specifically, we use a proposal $g(\phi'|\phi)$ which depends only on one variable selected randomly. Once this variable, say ϕ_i, has been chosen, we use a proposal proportional to the Boltzmann factor of the local term, namely

$$g(\phi_i) = C^{-1} e^{-\beta v(\phi_i)} \tag{5.102}$$

where C is a normalization constant. Replacing this proposal in the detailed balance condition (5.17), we obtain

$$e^{-\beta v(\phi_i')} h(\phi'|\phi) e^{-\beta v(\phi_i)} e^{-\beta \mathcal{H}_I(\phi)} = e^{-\beta v(\phi_i)} h(\phi|\phi') e^{-\beta v(\phi_i')} e^{-\beta \mathcal{H}_I(\phi')} \tag{5.103}$$

or

$$h(\phi'|\phi) e^{-\beta \mathcal{H}_I(\phi)} = h(\phi|\phi') e^{-\beta \mathcal{H}_I(\phi')}. \tag{5.104}$$

We have seen these functional equations earlier. They have the same form as the general detailed balance condition under the symmetric proposal (when the proposal functions g disappear from the equation). The novelty now is that only the interaction part of the Hamiltonian appears on both sides of this equation. Possible solutions are the Metropolis solution

$$h(\phi'|\phi) = \min(1, e^{-\beta \Delta \mathcal{H}_I}), \tag{5.105}$$

the Glauber solution

$$h(\phi'|\phi) = \frac{e^{-\beta \Delta \mathcal{H}_I}}{1 + e^{-\beta \Delta \mathcal{H}_I}}, \tag{5.106}$$

and so on.

Again, when computing $\Delta \mathcal{H}_I$, it is important to realize that this difference only depends on a reduced number of variables. For example, for the Φ^4 model, we would propose a value ϕ_i' sampled from the distribution

$$g(\phi_i') = C^{-1} e^{\frac{b}{2}\phi_i^2 - \frac{u}{4}\phi_i^4 + h\phi_i}. \tag{5.107}$$

Instead of using a rejection method as we proposed to sample (5.96), we can sample this new $g(\phi_i')$ using, for example, a numerical inversion algorithm because, at variance with (5.96), the parameters of (5.107), namely b, u, and h, do not vary after each step. The change in the interaction Hamiltonian needed for the acceptance step is then

$$\Delta \mathcal{H}_I = (\phi_i' - \phi_i) \sum_{\mu=1}^{d} \phi_{i_\mu}. \tag{5.108}$$

In general, a procedure similar to this one can be used when the Hamiltonian can be split into a sum of local terms plus an interaction term. One can choose the new value of the variable, independently of the old value, according to the distribution dictated by the local term. This proposal is then accepted with the Metropolis probability using only the interaction term.

5.7
Data Analysis: Problems around the Critical Region

Once we have developed our Monte Carlo code, the next logical step is to run it and to obtain some results of interest. However, this process is not without danger, as we will see using as an example the Ising model in the square lattice. In Figure 5.6,

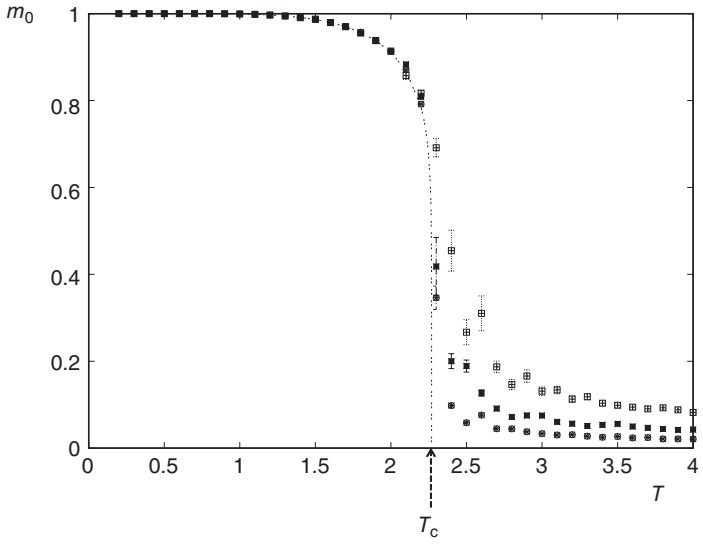


Figure 5.6 Spontaneous magnetization m_0 of the Ising model in the square lattice as a function of temperature T for different values of the system side $L = 20$, 40, 80 from top to bottom in the right-hand side of the figure. The dots have been obtained using the Metropolis algorithm described in the main text. The dashed line is Onsager's exact solution, as given by (5.52). We have set units such that $k = J = 1$.

we plot the spontaneous (at zero magnetic field) magnetization of the Ising model as obtained from our program for three different system sizes: $L = 20$, 40, and 80, as well as the exact solution given by Onsager (5.52).

From this figure, we notice clearly two things: (i) for a considerable range of temperatures, the numerical plots are well outside the theoretical result, and (ii) the error bars are not constant for all values of the temperature, but they are considerable larger near the critical region. The first discrepancy is due to finite-size effects. Concerning the magnitude of the error bars, we recall that the error in the sample average of the magnetization $\mathtt{m}$ is given by (5.14), which we rewrite here as

$$\epsilon[\mathtt{m}] = \frac{\sigma[\mathtt{m}]}{\sqrt{M}}\sqrt{2\tau_{\mathtt{m}} + 1} \tag{5.109}$$

where M is the number of points contributing to the sample average, $\sigma^2[\mathtt{m}] = \langle \mathtt{m}^2 \rangle - \langle \mathtt{m} \rangle^2$ is the variance of m, and $\tau_{\mathtt{m}}$ is the correlation time associated with the normalized correlation function $\rho_{\mathtt{m}}$. Certainly, the errors decrease by increasing the number of measurements, M, and we can ask which one of the two factors, $\sigma[\mathtt{m}]$ or $\tau_{\mathtt{m}}$, is responsible for the observed error increase around the critical region. The not-so-encouraging answer is that both factors contribute. Around the critical region one observes (i) an increase of fluctuations $\sigma[\mathtt{m}]$, and (ii) an increase of the correlation time $\tau_{\mathtt{m}}$ due to the so-called critical slowing down. Finite-size effects, increase of fluctuations, critical slowing down, and thermalization are the major

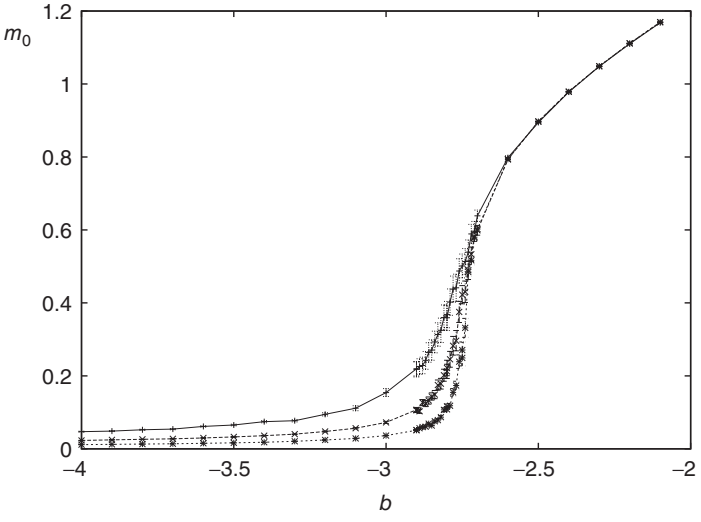


Figure 5.7 Spontaneous magnetization of the Φ^4 model in the square lattice as a function of the parameter b for $u = 1$ and different values of the system side $L = 20, 40, 80$ (from bottom to top lines). The data have been generated using the heat-bath Monte Carlo method explained in the main text.

points of concern in many Monte Carlo simulations. We now discuss these points separately. First of all, we must say that these problems are not specific to the Ising model. They can also be observed in the numerical simulations of the Φ^4 model (Figure 5.7) and of every other system that we simulate near a phase transition.

5.7.1 Finite-Size Effects

Finite-size effects appear as the discontinuities and mathematical singularities of a true phase transition requires of the thermodynamic limit $N \to \infty$. For example, Onsager's solution (5.52) implies a discontinuity of the derivative of $m_0(T)$ at $T = T_c$. In fact, the theory of phase transitions predicts that the asymptotic behavior near the critical point is

$$m_0(T) \sim |1 - T/T_c|^{\beta}, \quad \text{for } T \leq T_c \tag{5.110}$$

with a value of the "critical exponent" $\beta = 1/8$ for the 2-D Ising model. Similarly, it predicts that the magnetic susceptibility (5.49) diverges at the critical point as

$$\chi_T(T) \sim |1 - T/T_c|^{-\gamma} \tag{5.111}$$

with a value of the critical exponent $\gamma = 7/4$, for the 2-D Ising model, implying, formally, that the magnetic susceptibility diverges at $T = T_c$ and, according to (5.49), that the variance of the order parameter greatly increases near T_c. These are examples of the typical non-analytical behavior that occurs at the critical point. Strictly speaking, however, a non-analytical behavior cannot appear in a

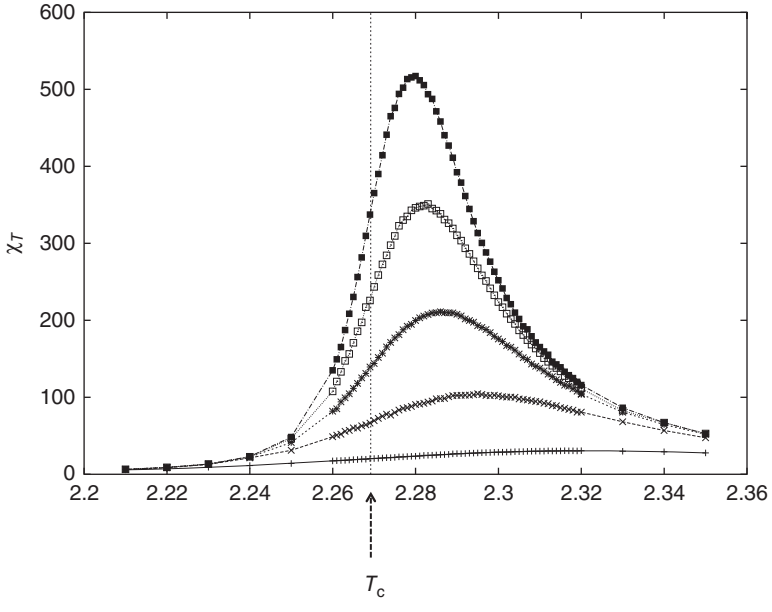


Figure 5.8 Magnetic susceptibility χ_T for the 2-D square Ising model computed from (5.49) using the Metropolis algorithm for system sides $L = 40, 80, 120, 160, 200$. The maximum value increases for increasing L.

finite system[11]. What it is observed in the simulations, though, is a big increase of the susceptibility in the neighborhood of the critical region (Figure 5.8). As the system size N increases, the maximum in the susceptibility grows higher and its location approaches the true critical temperature T_c of the infinite system.

It can be understood intuitively why finite-size effects will be more important near a second-order phase transition. In this situation, the correlation length, which measures the linear range over which spins at different sites of the lattice are correlated, diverges (in an infinite system) with a power-law singularity

$$\xi(T) \sim \left|1 - T/T_c\right|^{-\nu} \tag{5.112}$$

where $\nu = 1$ for the 2-D Ising model. For a finite system, the correlations cannot extend beyond the system side and we must have $\xi \sim L$. The theory of finite size tells us exactly how (and why!) the averages of interest behave. The basic idea is that now the magnetization becomes a *homogeneous* function of ξ and the system side L, that is, $m(\xi, L) = \xi^x \tilde{m}(\xi/L)$. The unknown exponent x is obtained by demanding that in the infinite system, and close enough to the critical point, one recovers the known behavior given by (5.110). This implies that the function $\tilde{m}(z)$ takes a finite

11) Think in terms of our simulation. It is impossible that χ_T, as given by (5.49), is equal to ∞ anywhere as the fluctuations of the order parameter are bounded, $\mathrm{m} < 1$, and we never divide by 0.

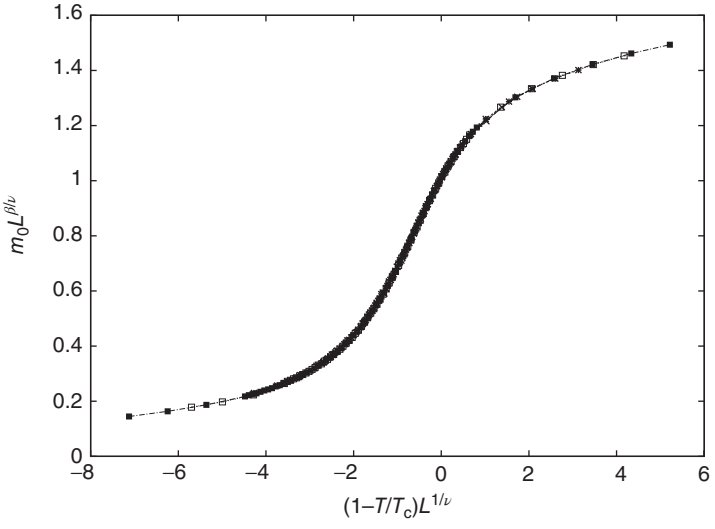


Figure 5.9 Check of the scaling relation with system size of the magnetization. Data for $L = 40, 80, 120, 160, 200$ have been collapsed onto the same curve using the rescaling of axes as indicated.

limit when $z \to 0$, and then

$$m_0(T) = \lim_{L\to\infty} m(T, L) = \xi^x \tilde{m}(0) \sim \left[|1 - T/T_c|^{-\nu}\right]^x \sim |1 - T/T_c|^{-x\nu}. \quad (5.113)$$

Compared to (5.110), one concludes $\beta = -x\nu$, and then the prediction for the magnetization near the critical point for a finite system is

$$m_0(T, L) = \xi^{-\beta/\nu} \tilde{m}(\xi/L) = L^{-\beta/\nu} \overline{m}[(1 - T/T_c)L^{1/\nu}]. \quad (5.114)$$

The typical way of checking this scaling behavior is to plot $m_0(T, L)L^{\beta/\nu}$ versus the rescaled variable $(1 - T/T_c)L^{1/\nu}$ (Figure 5.9). One can use as well the corresponding scaling relations for the specific heat and the susceptibility

$$c(T, L) = L^{\alpha/\nu} \overline{c}[(1 - T/T_c)L^{1/\nu}], \quad (5.115)$$

and

$$\chi_T(T, L) = L^{\gamma/\nu} \overline{\chi}[(1 - T/T_c)L^{1/\nu}]. \quad (5.116)$$

The scaling relation for χ_T has been checked in Figure 5.10. We have to say that the quality of the scaling observed in Figures 5.9 and 5.10 is not representative of many simulations in other systems. When plotting the figures with the rescaled data, we have used the *known* values for the critical temperature T_c and the critical exponents ν, β, and γ for the 2-D Ising model. If, as it is usually the case, the critical temperature and the critical exponents are not known, but their knowledge is the ultimate goal of our simulation, this procedure implies a three-parameter fit which is rather difficult to carry out in practice. The fitting has another important problem, namely, that the scaling relations, such as (5.114)–(5.116) and others, are

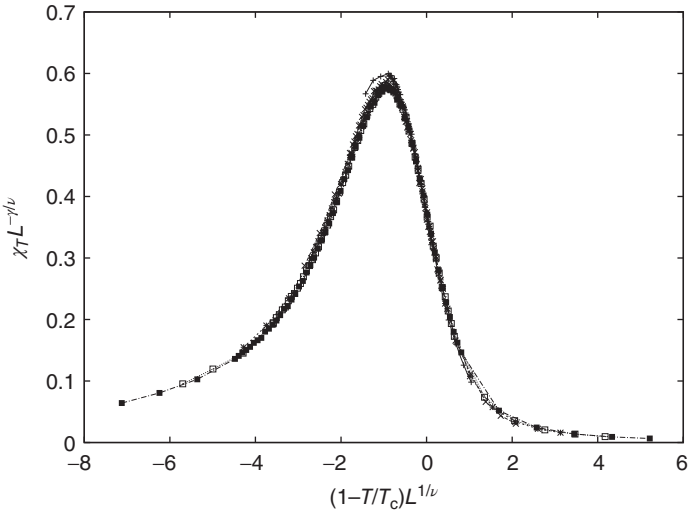


Figure 5.10 Check of the scaling relation with system size of the susceptibility. Data for $L = 40, 80, 120, 160, 200$ have been collapsed onto the same curve using the rescaling of axes as indicated.

only valid *asymptotically* for large L and close enough to T_c. What is meant by "large L" and "sufficiently close to T_c" is something that cannot be asserted before we do the simulations. If we add on top of all the large errors that occur near the critical region, it is not strange that some disputes appear continuously about the correct values of the critical exponents for different models of interest in the literature.

Of particular interest for the analysis of the data is the finite-size scaling behavior of the fourth-order cumulant, defined as the ratio of moments of the magnetization m defined as

$$U_4(T, L) \equiv 1 - \frac{\langle \mathrm{m}^4 \rangle}{3\langle \mathrm{m}^2 \rangle^2} = \overline{U}_4[(1 - T/T_c)L^{1/\nu}]. \tag{5.117}$$

As $U_4(T_c, L) = \overline{U}_4(0)$, a constant, the critical temperature T_c can be then determined as the common intersection of the $U_4(T, L)$ curves for different values of L (Figure 5.11). Once T_c has been determined by this procedure, we can use the fits to determine the values of the critical exponent ν and use those values in the analysis of the curves for the magnetization, susceptibility, and so on. However, be aware that these are not easy fits and it is essential to have a precise determination of the errors, including those due to the large autocorrelation times at the critical point.

5.7.2
Increase of Fluctuations

The variance $\sigma^2[m]$ measures the fluctuations of m. As we have discussed, the fluctuations increase at the critical region of a phase transition. This result can be

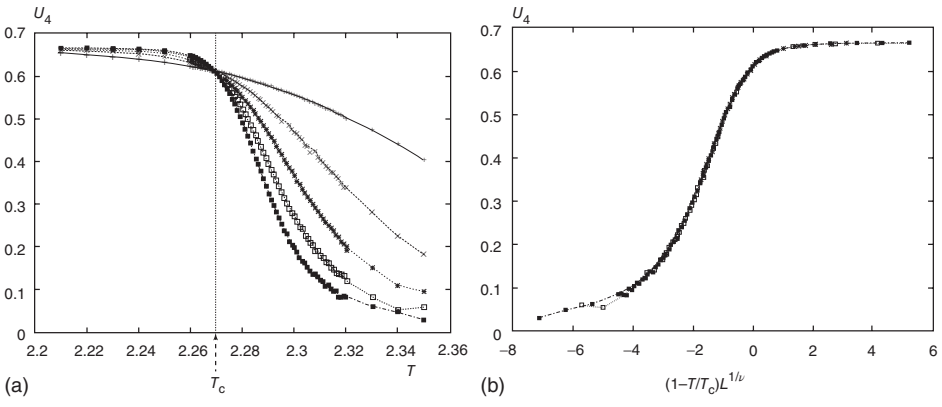


Figure 5.11 Behavior of the ratio of moments U_4 defined in (5.117) coming from numerical simulations of the 2-D square Ising model for system sizes $L = 40, 80, 120, 160, 200$. The common crossing of the curves for different values of L indicates the critical temperature T_c (a). (b) The data have been collapsed onto the same curve using the rescaling of the horizontal axis as indicated.

obtained from the relation between the fluctuations and some response functions, such as (5.49) and (5.51).

Near a critical point, the isotherm becomes flat and hence some of these derivatives become formally infinite and have, therefore, associated fluctuations of a very large amount. This increase in fluctuations at the critical point is the origin of the so-called critical opalescence that can be observed with the naked eye and constitutes one of the trademarks of critical points. However, from the numerical point of view, this is a disaster. If the fluctuations, as measured by $\sigma[m]$, increase, so does the error $\epsilon[m]$. The theory of finite-size scaling that we have briefly reviewed tells us that fluctuations of the order parameter behave at the critical region as dictated by (5.116). Therefore, exactly at $T = T_c$, it is $\chi_T(T_c, L) = L^{\gamma/\nu}\overline{\chi}(0)$. If we know the value of T_c, we can fit this expression to obtain γ/ν. There is another way in which we can proceed. As shown in Figure 5.8, the susceptibility $\chi_T(T, L)$ becomes a maximum at a location $T_c(L)$. By finding the maximum of (5.116), we can prove that the location of this maximum varies as

$$T_c(L) = T_c + a_1 L^{-1/\nu} \tag{5.118}$$

where a is a constant. From here, we deduce that a plot of $T_c(L)$ versus $L^{-1/\nu}$ must yield a straight line whose Y-intercept at the origin is precisely T_c. This can provide a confirmation of the values of T_c and ν obtained by other fits, such as the ones coming from the fourth-order cumulant.

5.7.3 Critical Slowing Down

As well as the fluctuations, the correlation time τ_m also diverges at the critical point! It is known that the *real* dynamics of a physical system slows down at the

critical region (again, a fact that can be observed experimentally). However, we are not using here the real (say Hamiltonian) dynamics, but we have introduced (within some arbitrariness) a convenient Markov chain that allows us to generate representative configurations at equilibrium. Does this stochastic dynamics also suffer from critical slowing down? The answer is yes, as shown by the numerical data and supported by a simple calculation. We had derived in (4.80) an exact expression for the value of the correlation function $\rho_G(1)$ after one elementary proposal–acceptance step. We combine this with the approximate expression for the correlation time (4.26) to obtain

$$\tau_{\mathfrak{m}} \approx \frac{2\sigma^2[\mathfrak{m}]}{\int dx\, dy\, h(x|y)g(x|y)[\mathfrak{m}(x) - \mathfrak{m}(y)]^2 f_{\tilde{\mathbf{x}}}(y)}. \tag{5.119}$$

Consider, for the sake of clarity, the Metropolis algorithm for the Ising model, and we want to compute the correlation time of the magnetization (5.46). We know that the proposal for change $x \to y$ is to reverse the sign of a single spin. This yields a modification of $\mathfrak{m}(x) - \mathfrak{m}(y) = \frac{\pm 2}{N}$. After squaring, it can be taken out of the integral and we are left with $\int dx\, dy\, h(x|y)g(x|y)f_{\tilde{\mathbf{x}}}(y)$, which is nothing but the average acceptance probability ϵ. We get finally

$$N^{-1}\tau_{\mathfrak{m}} = \frac{N\sigma^2[m]}{2\epsilon} = \frac{kT\chi_{_T}}{2\epsilon}. \tag{5.120}$$

$N^{-1}\tau_{\mathfrak{m}} \equiv \tau_{\mathfrak{m}}^{\mathrm{MCS}}$ is the autocorrelation time in units of updates per spin, or the MCS. As the average acceptance probability does not decrease near zero at the critical point, it follows that, within this approximation, the critical behavior of the correlation time $\tau_{\mathfrak{m}}^{\mathrm{MCS}}$ is that of the magnetic susceptibility $\chi_{_T}$. As this diverges at the critical point, it turns out that $\tau_{\mathfrak{m}}^{\mathrm{MCS}}$ diverges as well. As explained in the previous section, for a finite system of linear side L, it cannot really diverge to infinity, but it grows as a power of the system side

$$\tau_{\mathfrak{m}}^{\mathrm{MCS}}(T = T_{\mathrm{c}}) \sim L^z. \tag{5.121}$$

The simple expression (5.120) we have derived here would tell us that $z = \gamma/\nu$, the exponent giving the divergence of the susceptibility with system size. Intensive numerical simulations using the Metropolis algorithm suggest that the dynamical critical exponent takes a value $z \approx 2$ for the 2-D Ising model. As the fluctuations increase at the critical point as $\chi_{_T} \sim L^{\gamma/\nu}$, it turns out that the error (5.109) at the critical point increases with system size as $\epsilon[m] \sim L^{(z+\gamma/\nu)/2}$ with $(z + \gamma/\nu)/2 \approx 1.9$. Roughly speaking, for the 2-D Ising model, the errors in the Metropolis algorithm near the critical point increase by a factor of 4 every time we double the system linear side, if we keep the number of measurements M constant.

While the divergence of the fluctuations (e.g., susceptibility) near a critical point is intrinsic to the model under study, the divergence of $\tau_{\mathfrak{m}}$ and other correlation times is a property of the numerical method we use to generate the representative configurations. Therefore, some Monte Carlo schemes have a smaller correlation time than others. It is then important to compare the different possibilities available and to tune up the parameters of the numerical method, whenever possible, in

order to obtain the smallest correlation time. As far as the Ising model is concerned, the best algorithms use collective updates in which many spins are changed at once. In this way, it is possible to reduce the critical exponent z to a value close to 0. This, however, requires extreme care in the choice of the proposal and acceptance probabilities. They will be discussed briefly in Appendix D.

5.7.4 Thermalization

We will be very brief here. We just want to remind the reader that the Boltzmann factor is sampled only asymptotically by the Monte Carlo algorithm. Therefore, we need to make sure that we are in the regime where all generated configurations follow the distribution (5.3). We have already discussed that one way of checking that this is the case is to check the result (5.24). More sophisticated tests include computing the nonlinear relaxation function and checking whether it has decayed to a value close to 0. We refer the reader to Section 4.7 for a discussion of this point.

Further Reading and References

Concerning the applications to lattice Ising-type models, the books by M.E.J. Newman and G.T. Barkema [13] and by D.P. Landau and K. Binder [14] give further details about data analysis using finite-size scaling techniques. The second book also offers a good summary of techniques and results for off-lattice continuous models such as Lennard–Jones and hard spheres. This topic is also covered in the classic book by Allen and Tildesley [15].

Exercises

1) Consider a hard-sphere model with a gravitational potential $-mgz$ acting on each sphere. Use the Monte Carlo algorithm to compute the density profile $\rho(z)$ as a function of height.
2) Prove that, in the Ising model, if we select randomly the spin to be updated, then the proportion of nodes that are not selected after N trials tends, for large N, to the value $1/e \approx 37\%$.
3) Compute numerically the autocorrelation times for the magnetization and the energy for the 2-D Ising model using Metropolis algorithm, as a function of temperature T for systems of size $L \times L$ with $L = 20, 40, 60, 80$. Compare the result given by the analysis of the temporal series (using, e.g., the method given in Appendix C) with the approximate value (4.26).
4) Run the Monte Carlo algorithm for the Ising model at exactly the critical temperature $T_c = 1/\log\sqrt{1+\sqrt{2}} \approx 2.2691853\ldots$ and compute the magnetization, the susceptibility, and the correlation time of the magnetization for a square lattice of systems size $L \times L$ with $L = 20, 40, 80, 160, 320$ (and larger

if your computer allows you to do so). Check that all these quantities behave proportionally to a power of the system size L, and compute in each case the exponent of the power law. Do not forget to perform a proper analysis, and give meaningful error bars for each exponent.

5) Modify the program for the 2-D Ising model including the so-called Moore neighborhood, in which each spin interacts with its four nearest neighbors and the four next-nearest neighbors.
6) Run the heat-bath algorithm for the Ising model in the 3-D cubic lattice. Use the finite-size scaling technique analysis of the fourth-order cumulant and determine the critical temperature (best estimate in the literature $T_c \approx 4.5115$).
7) In the so-called q-state Potts model, in every site of the lattice, the variable s_i can take any integer value between 1 and q (the case $q = 2$ is isomorphic to the Ising model). The interaction energy between two nearest neighbors s_i, s_j is equal to 0 if $s_i \neq s_j$ and equal to -1 if $s_i = s_j$. If we want to implement the Metropolis algorithm, what is a reasonable proposal probability in this model? What is the corresponding acceptance probability? Run the program for $q = 3$ and compute the order parameter, defined as

$$\mathrm{m} = \sqrt{\frac{1}{q-1}\sum_{k>\ell}(x_k - x_\ell)^2}$$

where $x_k, k = 1, \ldots, q$, is the proportion of sites occupied by a value $s_i = k$. Determine the critical temperature.
8) An elegant way of implementing a Monte Carlo method for the Φ^4 model is that of Bruce [16]. In his approach, the proposal $g(x|y) \equiv g(\phi'_i)$ is also independent of the old configuration y (in the same vein as in heat bath), but $g(\phi'_i)$ is chosen to be the sum of two Gaussians that best approximate the actual local distribution of the field ϕ'_i:

$$g(\phi'_i) = \frac{1}{2}\left[\frac{1}{\sigma_1\sqrt{2\pi}}e^{-\frac{(\phi'_i-\mu_1)^2}{2\sigma_1^2}} + \frac{1}{\sigma_2\sqrt{2\pi}}e^{-\frac{(\phi'_i-\mu_2)^2}{2\sigma_2^2}}\right],$$

and $\sigma_1, \sigma_2, \mu_1, \mu_2$ are determined self-consistently during the simulation. Use the detailed balance condition to determine a possible acceptance probability for this method.
9) Prove (5.118) for the location of the maximum susceptibility according to the theory of finite-size scaling.
10) Implement a simple Metropolis algorithm for the 2-D square lattice Φ^4 model in which the proposal changes the scalar variable $\phi_i \to \phi'_i \in (\phi_i - \Delta, \phi_i + \Delta)$. Run the program at the approximate value of the critical point ($u = 1, b \approx -2.735$) and determine the optimal value for Δ as the one that minimizes the correlation time of the magnetization.
11) Write the Hamiltonian (5.78) of the Φ^4 model with $H = 0$ in terms of the rescaled variables $s_i = \frac{\Phi_i}{\sqrt{\frac{B-2dK}{U}}}$, and show that in the limit

$$B \to \infty, U \to \infty, \frac{B}{U} \to \text{ constant}, \quad \beta J = K\frac{B}{U},$$

one obtains for the Gibbs factor

$$e^{-\beta\mathcal{H}([S])} = \prod_{i=1}^{N} \left[\delta(s_i^2 - 1)\right] \exp\left[\beta J \sum_{\langle i,j\rangle} s_i s_j\right].$$

Prove that this is equivalent to writing

$$\mathcal{H}([S]) = -J \sum_{\langle i,j\rangle} s_i s_j, \quad s_i = \pm 1,$$

which is nothing but the Hamiltonian of the Ising model.

6
Introduction to Stochastic Processes

In this chapter, we review the basic concepts of what a stochastic process is. Our aim is not to be rigorous on the mathematical side but rather to focus on the physical insights behind the concepts. The name ''stochastic process'' is usually associated with a trajectory in phase space which is random enough to demand a probabilistic description. A paradigmatic example is that of the Brownian motion.

6.1
Brownian Motion

The botanist Robert Brown discovered in 1827 that particles in a pollen in suspension execute random movements which he even interpreted initially as some sort of life. It is not so well known that L. Boltzmann knew as early as 1896 the reason for this erratic movement when he wrote ''... very small particles in a gas execute motions which result from the fact that the pressure on the surface of the particles may fluctuate.'' However, it was A. Einstein in 1905 who successfully introduced the first mathematical treatment of the erratic movement of the Brownian particles. Rather than focusing on the (complicated) trajectory of a single particle, Einstein introduced a probabilistic description valid for an ensemble of Brownian particles. First, Einstein introduced the concept of a coarse-grained description defined by a time τ such that different parts of the trajectory separated by a time τ or larger can be considered independent. No attempt is made to characterize the dynamics at a timescale smaller than this coarse-grain time τ. Instead, we consider snapshots of the system taken at time intervals τ (Figure 6.1).

The second concept, probabilistic in nature, introduced by Einstein is that of the probability density function, $f(\vec{\Delta})$, for the three-dimensional distance $\vec{\Delta} = (\Delta_x, \Delta_y, \Delta_z)$ traveled by the Brownian particle in a time interval τ. One could assume, for instance, a Gaussian form for $f(\vec{\Delta})$, but this is not necessary. In fact, the only assumption one needs to make about the function $f(\vec{\Delta})$ (besides the general condition of nonnegativity and normalization) comes from the fact that the collisions of the fluid molecules and the Brownian particle occur with the same

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

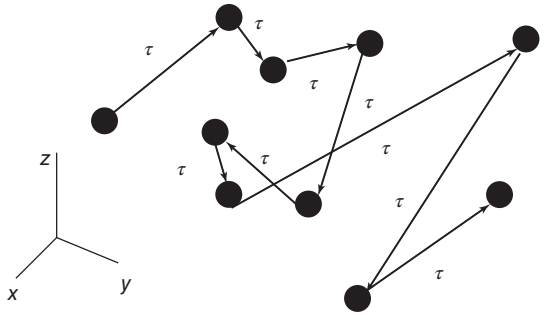


Figure 6.1 Schematic representation of the movement of a Brownian particle.

probability in any direction[1]. The absence of preferred directions translates to a symmetry condition for $f(\vec{\Delta})$, as

$$f(-\Delta) = f(\Delta). \tag{6.1}$$

The third step in this description is to consider an ensemble of N Brownian particles in a large enough system. Also, we focus on spatial scales that are much larger than the size of a Brownian particle so that we can define a density of the particles $n(\vec{x}, t)$ such that $n(\vec{x}, t)d\vec{x}$ is the number of particles in the interval $(\vec{x}, \vec{x} + d\vec{x})$ at time t. From the assumption that the parts of the trajectories separated a time interval τ are statistically independent, it follows that the number of particles at location $\vec{x}$ at time $t + \tau$ will be given by the number of particles at location $\vec{x} - \vec{\Delta}$ at time t multiplied by the probability that the particle jumps from $\vec{x} - \vec{\Delta}$ to $\vec{x}$, which is $f(\vec{\Delta})$, and integrated for all the possible $\vec{\Delta}$ values: that is

$$n(\vec{x}, t + \tau) = \int_{\mathbb{R}^3} n(\vec{x} - \vec{\Delta}, t) f(\vec{\Delta})\, d\vec{\Delta}. \tag{6.2}$$

This is the basic evolution equation for the number density $n(\vec{x}, t)$. From the technical point of view, it is a continuity equation whose interpretation is that Brownian particles can neither be created, nor can they disappear as a result of the collisions with the fluid molecules. By Taylor expansion of the above expression and making use of the symmetry relation (6.1), one gets the diffusion equation as

$$\frac{\partial n}{\partial t} = D\nabla^2 n \tag{6.3}$$

where the *diffusion coefficient* D is given in terms of the pdf $f(\vec{\Delta})$ by

$$D = \frac{1}{2\tau} \int_{-\infty}^{\infty} \vec{\Delta}^2 f(\vec{\Delta}) d\vec{\Delta} = \frac{\langle \vec{\Delta}^2 \rangle}{2\tau}. \tag{6.4}$$

If the initial condition is such that all N particles are located at the origin, that is, $n(\vec{x}, t = 0) = N\delta(\vec{x})$, the solution of the diffusion equation is

$$n(\vec{x}, t) = \frac{N}{(4\pi D t)^{3/2}} \mathrm{e}^{-\vec{x}^2/4Dt} \tag{6.5}$$

1) We also disregard the effect of gravity in the Brownian particle, which would lead to a preferred direction in the movement.

from where it follows that the average position of the Brownian particle is $\langle \vec{x}(t) \rangle = 0$ and that the average square position increases linearly with time, namely

$$\langle \vec{x}(t)^2 \rangle = 6Dt. \tag{6.6}$$

This prediction has been successfully confirmed in experiments and contributed to the acceptance of the atomic theory.

However successful Einstein's approach was, it is very phenomenological and cannot yield, for instance, an explicit expression that allows the calculation of the diffusion coefficient in terms of microscopic quantities. Langevin (1908) initiated a different approach which, in some ways, can be considered complementary to the previous one. In his approach, Langevin focused on the trajectory of a single Brownian particle and wrote down Newton's equation Force = mass × acceleration. The trajectory of the Brownian particle is highly erratic and therefore its description would demand a peculiar kind of force. Langevin considered two types of forces acting on the Brownian particle: the usual friction forces which, according to Stokes law, would be proportional to the velocity, and a sort of "fluctuating" force $\vec{\xi}(t)$, which represents the "erratic" force that comes from the action of the fluid molecules on the Brownian particle. The equation of motion becomes then

$$m\frac{d\vec{v}}{dt} = -6\pi\eta a\vec{v} + \vec{\xi} \tag{6.7}$$

where η is the viscosity coefficient and a is the radius of the Brownian particle (which is assumed to be spherical). Multiplying both sides of (6.7) by $\vec{x}$, one gets

$$\frac{m}{2}\frac{d^2\vec{x}^2}{dt^2} - m\left(\frac{d\vec{x}}{dt}\right)^2 = -3\pi a\eta\frac{d\vec{x}^2}{dt} + \vec{x}\cdot\vec{\xi}. \tag{6.8}$$

Langevin made two assumptions about the fluctuating force $\vec{\xi}(t)$: that it has mean 0 (collisions do not push the Brownian particle in any preferred direction) and that it is uncorrelated to the actual position of the Brownian particle (the action of the molecules of fluid on the Brownian particle is the same no matter the location of the Brownian particle), that is

$$\begin{aligned} \langle \vec{\xi}(t) \rangle &= 0, \\ \langle \vec{x}\cdot\vec{\xi} \rangle &= \langle \vec{x} \rangle \cdot \langle \vec{\xi} \rangle = 0. \end{aligned} \tag{6.9}$$

Taking the averages in (6.8) with respect to all realizations of the random force $\vec{\xi}(t)$ and using the previous conditions on $\vec{\xi}(t)$, one gets

$$\frac{m}{2}\frac{d^2\langle \vec{x}^2 \rangle}{dt^2} = m\langle \vec{v}^2 \rangle - 3\pi a\eta\frac{d\langle \vec{x}^2 \rangle}{dt} \tag{6.10}$$

which is an equation for the average square position of the Brownian particle. Langevin assumed that we are now in a regime in which thermal equilibrium between the Brownian particle and the surrounding fluid has been reached. In particular, this implies that, according to the equipartition theorem, the average kinetic energy of the Brownian particle is $\langle m\vec{v}^2/2 \rangle = 3kT/2$ (k is Boltzmann's constant and T is the fluid temperature). One can now solve (6.10) and find that, after some transient time, the asymptotic mean square displacement is given by

$$\langle \vec{x}^2 \rangle = \frac{kT}{\pi \eta a} t. \tag{6.11}$$

This is nothing but Einstein's diffusion law, but now we have an explicit expression for the diffusion coefficient in terms of other macroscopic variables:

$$D = \frac{kT}{6\pi \eta a}. \tag{6.12}$$

Langevin's random force $\vec{\xi}(t)$ is an example of a stochastic process. It is time to proceed to a more precise definition of a stochastic process. The natural machinery is that of probability theory.

6.2 Stochastic Processes

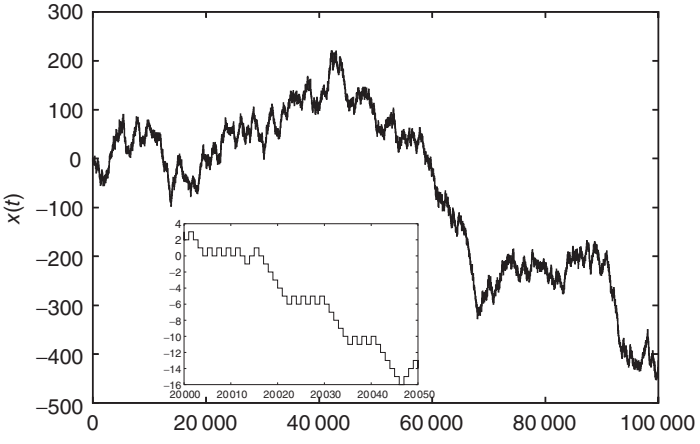
In Chapter 1, we introduced the concept of a random variable $\hat{x}$ resulting from a probabilistic experiment. We now define a stochastic process as a family $\hat{x}(t)$ of random variables depending on some continuous, real parameter t. In most applications, t is a physical time and the stochastic process can be thought as performing multiple probabilistic experiments one at each time instant. The trajectory followed by the system depends on the outcome of each probabilistic experiment. As a consequence, the knowledge of the initial condition x_0 at time t_0 is not enough to determine the position in phase space of the system at a later time t_1. Instead, the trajectory, and therefore the final state of the system, acquires a probabilistic nature. To fully determine the final state, it is also necessary to know the outcome of all the successive probabilistic experiments between the initial time t_0 and the final time t_f. In fact, each possible set of successive outcomes determines a possible trajectory for the system, all starting at x_0 at time t_0 but ending at different locations at time t_f. The stochastic process can be seen as the collection of all these possible trajectories.

Arguably, the most well-known example of a stochastic process is that of the random walk. The probabilistic experiment is now a series of binary results representing, for instance, the outcome of repeatedly tossing a coin:

$$(0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 1, 1, 1, 0, 0, 1, \dots)$$

where 1 means ''heads'' and 0 means ''tails.'' Consider that the tossing takes place at given times $0, \tau, 2\tau, \dots$. To the outcome of this set of probabilistic experiments, we associate a one-dimensional function $x(t)$ which starts at $x(0) = 0$ and that moves to the left (right) at time $k\tau$ an amount a ($-a$) if the kth result of the tossed coin was 0 (1). In the intermediate times between two consecutive tossings, namely in the times between $k\tau$ and $(k+1)\tau$, the system just remains in the same location. Figure 6.2 shows a typical trajectory.

What does one mean by characterizing a stochastic process? Since it is nothing but a continuous family of random variables, a stochastic process will be completely characterized when we give the joint probability density function for the arbitrary



Por qué no hay un estacionario alrededor del 0?

Figure 6.2 Example of a random walk trajectory. We have taken $\tau = a = 1$ and plotted the resulting trajectory after a large number (10^5) of steps. In the inset, we see the fine detail with the discrete jumps occurring at times which are multiples of τ.

set $\{\hat{x}(t_1), \hat{x}(t_2), \ldots, \hat{x}(t_m)\}$, that is, when we give the function $f(x_1, \ldots, x_m; t_1, \ldots, t_m)$ for an arbitrary m. This function is such that

$$f(x_1, \ldots, x_m; t_1, \ldots, t_m)dx_1 \ldots dx_m \tag{6.13}$$

represents the probability that the random variable $\hat{x}(t_1)$ takes values in the interval $(x_1, x_1 + dx_1)$, the random variable $\hat{x}(t_2)$ takes values in the interval $(x_2, x_2 + dx_2)$, and so on.[2] We can see here a generalization of the successions of random variables presented in 1.4. As t is a continuous variable, we have here formally a nonnumerable, infinite number of random variables $\hat{x}(t)$. However, when we extract from this a finite set $\hat{x}(t_1), \ldots, \hat{x}(t_m)$, we can use the same definitions and results as in the case of a succession of random variables. For example, we extend now the Markov property to a stochastic process.

A stochastic process is said to be a Markov process if the rather general conditional probability

$$f(x_m; t_m | x_1, \ldots, x_{m-1}; t_1, \ldots, t_{m-1}) \equiv \frac{f(x_1, \ldots, x_m; t_1, \ldots, t_m)}{f(x_1, \ldots, x_{m-1}; t_1, \ldots, t_{m-1})} \tag{6.14}$$

is equal to the two-times conditional probability

$$f(x_m; t_m | x_{m-1}; t_{m-1}) \equiv \frac{f(x_{m-1}, x_m; t_{m-1}, t_m)}{f(x_{m-1}; t_{m-1})} \tag{6.15}$$

for all times $t_m > t_{m-1} > \ldots t_1$. This Markov property, which, loosely speaking, means that the probability of a future event depends only on the present state of the system and not on the way it reached its present situation, allows us to compute

2) In a different language, we can say that a complete characterization of the trajectory is obtained by giving the functional probability density function $f([x(t)])$.

the m-times pdf as

$$f(x_1, \dots, x_m; t_1, \dots, t_m) = f(x_m; t_m | x_{m-1}; t_{m-1}) f(x_{m-1}; t_{m-1} | x_{m-2}; t_{m-2}) \dots f(x_2; t_2 | x_1; t_1) f(x_1; t_1). \tag{6.16}$$

The random walk constitutes an example of a Markov process, as the probability of having a particular value of the position at time $(k+1)\tau$ depends only on the particle's location at time $k\tau$ and not on the way it got to this location.

A particular case of a Markov process is a completely independent process in which an arbitrary set of random variables at different times are independent, and then we are able to write

$$f(x_1, \dots, x_m; t_1, \dots, t_m) = f(x_1; t_1) f(x_2; t_2) \dots f(x_m; t_m). \tag{6.17}$$

Another important example of a random process is that of a Gaussian process in which the m-times pdf admits an explicit form generalizing (1.80), namely

$$f(x_1, \dots, x_m; t_1, \dots, t_m) = \sqrt{\frac{|A|}{(2\pi)^m}} \exp\left[-\frac{1}{2}\sum_{i,j=1}^{m}(x_i - b_i)A_{ij}(x_j - b_j)\right] \tag{6.18}$$

where

$$\begin{aligned} b_i &= \langle x(t_i)\rangle, \\ (A^{-1})_{ij} &= \langle x(t_i)x(t_j)\rangle - \langle x(t_i)\rangle\langle x(t_j)\rangle. \end{aligned} \tag{6.19}$$

For Gaussian processes, the explicit form of the pdf (6.18) is rarely written down, rather the process is fully characterized by giving the mean value $\langle \hat{x}(t)\rangle$ and the correlation function $\langle \hat{x}(t)\hat{x}(t')\rangle$.

6.3 Stochastic Differential Equations

A stochastic differential equation is a differential equation that contains a stochastic process $\hat{\eta}(t)$: that is, an equation of the form

$$\frac{d\hat{x}(t)}{dt} = G(\hat{x}, t, \hat{\eta}(t)) \tag{6.20}$$

where G is a given function that depends, in general, on the variable $x(t)$, on the time t, and on the stochastic process $\hat{\eta}(t)$. A stochastic differential equation can be seen as a family of ordinary differential equations, one for each outcome of all the successive probabilistic experiments associated with the stochastic process $\hat{\eta}(t)$. As a consequence, for any given initial condition x_0 at time t_0, one has a family of possible trajectories. Therefore, $\hat{x}(t)$, which is the collection of all these possible trajectories, has to be viewed also as a stochastic process and this is why we label it with the ''hat'' symbol. However, $\hat{x}(t)$ is not an arbitrary stochastic process, rather it depends on $\hat{\eta}(t)$ in a specific manner determined by the stochastic differential equation, and, as a consequence, the statistical properties of $\hat{x}(t)$ depend on the statistical properties of $\hat{\eta}(t)$.

Strictly speaking, "solving the stochastic differential equation" means to provide the complete characterization of the stochastic process $\hat{x}(t)$, namely to give all the m-times pdfs $f(x_1, \dots, x_m; t_1, \dots, t_m)$, in terms of the statistical properties of $\hat{\eta}(t)$. However, one has to understand that a complete characterization of a general stochastic process implies the knowledge of a function of an arbitrary number of parameters and is very difficult to carry out in practice. On many occasions, one is happy if one can give just the one-time pdf $f(x; t)$ and the two-times pdf $f(x_1, x_2; t_1, t_2)$. In terms of those, it is possible to compute the trajectory averages

$$\langle \hat{x}(t)^n \rangle = \int_{-\infty}^{\infty} dx\, x^n f(x; t) \tag{6.21}$$

and the time correlations

$$\langle \hat{x}(t_1)\hat{x}(t_2) \rangle = \int_{-\infty}^{\infty} dx_1 \int_{-\infty}^{\infty} dx_2\, x_1 x_2\, f(x_1, x_2; t_1, t_2). \tag{6.22}$$

In general, the function G can depend on the stochastic process $\hat{\eta}(t)$ in an arbitrary way. However, many systems of interest can be described by stochastic differential equations in which $\hat{\eta}(t)$ appears linearly, namely

$$\frac{d\hat{x}}{dt} = q(\hat{x}) + g(\hat{x})\hat{\eta}(t). \tag{6.23}$$

This kind of stochastic differential equations, which are the only ones to be considered in this book, are called *Langevin equations*. In this case, the independent stochastic process $\hat{\eta}(t)$ is usually referred to as *noise*, a notation that comes from the early days of radio broadcasting when the random fluctuations in the electrical signals taking place in the emitter during the propagation in the atmosphere or at the receiver device led to noises that were actually heard on top of the radio emission. Following this notation, in (6.23), the term $g(\hat{x})\hat{\eta}(t)$ is referred to as the *noise term* or *diffusion term* whereas $q(\hat{x})$ is the *deterministic term* or *drift term*. One distinguishes the case in which the function $g(\hat{x})$ is a constant, in which case the noise is said to be *additive*. Otherwise, the noise is said to be *multiplicative*.

For the sake of simplicity in the notation, from now on we will drop the "hats" from the stochastic process and therefore we write the Langevin differential equation as

$$\frac{dx}{dt} = q(x) + g(x)\eta(t). \tag{6.24}$$

We have already encountered an example of Langevin differential equation in Section 6.1, (6.7), the equation introduced by Langevin himself to describe the movement of a Brownian particle. In this case, the independent stochastic process is the random force that acts on the Brownian particle and models the collisions of the water molecules and it appears in the equation as an additive noise, and the deterministic term is the drag induced by the water viscosity. And, somehow, we have already "solved" this Langevin equation when we determined some statistical properties of the movement of the Brownian particle, such as the mean square displacement.

6.4 White Noise

We proceed now to characterize in a more detailed way the stochastic process $\xi(t)$ that appears in the Langevin equation for the Brownian motion. To do this, we first start with the characterization of another process we have already encountered, the one-dimensional random walk. Starting at $x = 0$ at time $t = 0$, the location of the random walker after tossing the coin n times is given by the number of steps taken in the sense that x increases n_1 (number of "heads") minus the number of steps taken in the opposite sense n_0 (number of "tails"), $x(n\tau) = (n_1 - n_0)a = (2n_1 - n)a$. The probability of having n_1 "heads" after n throws is given by the binomial expression

$$P(n_1) = \binom{n}{n_1} 2^{-n}. \tag{6.25}$$

Therefore, the probability that the walker is at a location $x = ra$ after a time $t = n\tau$ is given by

$$P(x(n\tau) = ra) = \binom{n}{\frac{n+r}{2}} 2^{-n}. \tag{6.26}$$

From the above, it follows using (1.31)–(1.32)

$$\begin{aligned} \langle x(n\tau)\rangle &= 0, \\ \langle x(n\tau)^2\rangle &= na^2. \end{aligned} \tag{6.27}$$

As explained in Section 1.3 for $n \gg 1$, the binomial distribution can be approximated by a Gaussian distribution (de Moivre–Laplace theorem)

$$P(x(n\tau) \leq ra) = \frac{1}{2} + \operatorname{erf}\left(\frac{r}{\sqrt{n}}\right). \tag{6.28}$$

We now take the continuum limit defined by

$$n \to \infty,\ \tau \to 0,\ r \to \infty,\ a \to 0 \tag{6.29}$$

while preserving a finite value for

$$t = n\tau,\ x = ra,\ D = a^2/\tau. \tag{6.30}$$

In this limit, the random walk process is called the *Wiener process* $W(t)$ and (6.28) can be written as

$$P(W(t) \leq x) = \frac{1}{2} + \operatorname{erf}\left(\frac{x}{\sqrt{Dt}}\right) \tag{6.31}$$

which is the probability distribution function of a Gaussian variable with zero mean and variance Dt. The corresponding probability density function is

$$f(x;t) = \frac{1}{\sqrt{2\pi Dt}} \exp\left(-\frac{x^2}{2Dt}\right). \tag{6.32}$$

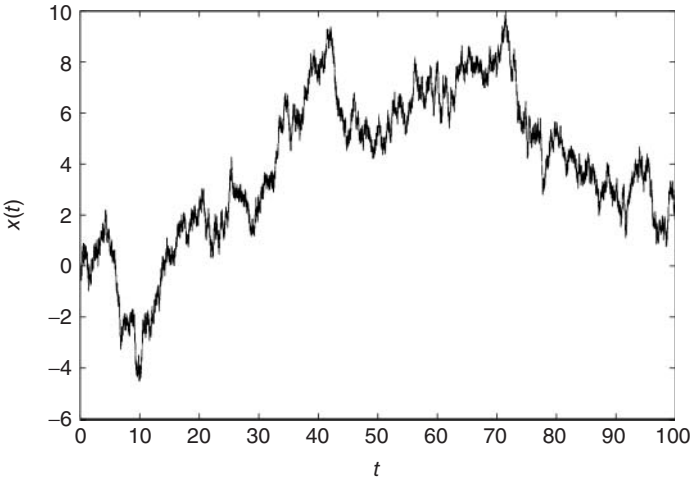


Figure 6.3 Typical realization of the Wiener process generated using the random walk with $\tau = 10^{-4}$ and $a = 10^{-2}$.

The Gaussian Wiener process inherits the Markovian character of the random walk process and can be characterized by giving its mean value and the two-times correlation function:

$$\langle W(t)\rangle = 0, \tag{6.33}$$

$$\langle W(t_1)W(t_2)\rangle = D\min(t_1, t_2). \tag{6.34}$$

A plot of a typical realization of the Wiener process is shown in Figure 6.3.

The random walk process was a sequence of step functions. As a consequence, the Wiener process is continuous but it does not have a well-defined first derivative[3]. Still, we will define now the *white noise* random process $\xi(t)$ as the derivative of the Wiener process. As we just mentioned that the Wiener process does not have a well-defined derivative, it is not surprising that the result depends on the way the derivative is performed. We first go back to the discrete random walk process $x(t)$ and define a new stochastic process w_ϵ as

$$w_\epsilon(t) = \frac{x(t+\epsilon) - x(t)}{\epsilon}. \tag{6.35}$$

A sketch of the process w_ϵ is given in Figure 6.4. When taking the continuum limit (6.29), $w_\epsilon(t)$ tends to a Gaussian process since it is a linear combination of Gaussian processes. Therefore, it is sufficiently defined by its mean and correlations:

$$\langle w_\epsilon(t)\rangle = 0, \tag{6.36}$$

3) In fact, it is a fractal of dimension 1/2.

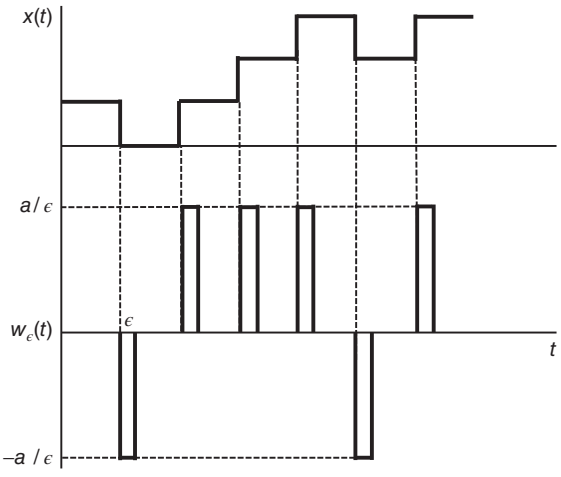


Figure 6.4 Random walk and its derivative.

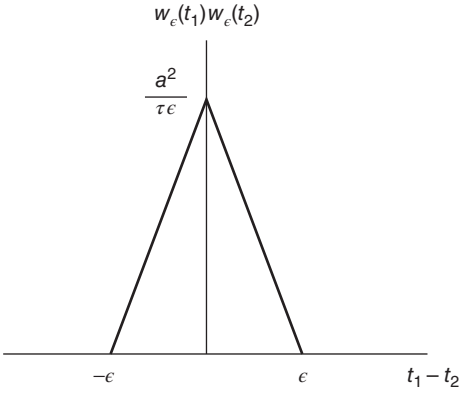


Figure 6.5 Correlation function for the derivative of the random walk process as given by (6.37).

$$\langle w_\epsilon(t_1)w_\epsilon(t_2)\rangle = \begin{cases} 0, & t_1 - t_2 < -\epsilon, \\ a^2/(\tau\epsilon)(1 + (t_1 - t_2)/\epsilon), & -\epsilon \le t_1 - t_2 \le 0, \\ a^2/(\tau\epsilon)(1 - (t_1 - t_2)/\epsilon), & 0 \le t_1 - t_2 \le \epsilon, \\ 0, & t_1 - t_2 > \epsilon. \end{cases} \tag{6.37}$$

The shape of the correlation function is shown in Figure 6.5.

In the limit $\epsilon \to 0$, the process $w(t) = \lim_{\epsilon\to 0} w_\epsilon(t)$ becomes the derivative of the random walk process and the correlation function (6.37) becomes a delta function

$$\langle w(t_1)w(t_2)\rangle = (a^2/\tau)\delta(t_1 - t_2). \tag{6.38}$$

If we go now to the limit defined by (6.29) and (6.30), the random walk tends to the Wiener process and its derivative can be written as $w(t) = D^{1/2}\xi(t)$, where $\xi(t)$ is a

Markovian, Gaussian process of zero mean and correlations

$$\langle \xi(t) \rangle = 0, \tag{6.39}$$

$$\langle \xi(t_1)\xi(t_2) \rangle = \delta(t_1 - t_2), \tag{6.40}$$

known as white noise. It is the derivative of the Wiener process (for $D = 1$):

$$\xi(t) = \frac{dW(t)}{dt}. \tag{6.41}$$

The white noise can be understood as a series of pulses each of which is very short but very intense, in a way that their effect is finite. The pulses are independent among them, which represents the perturbations acting on the system in random directions so that the average of all the perturbations is zero (6.39). The name comes from the fact that its power spectral density (the Fourier transform of the correlation function) is flat: that is, it is the same at all frequencies.

6.5 Stochastic Integrals. Itô and Stratonovich Interpretations

In the previous section, we introduced the white noise stochastic process as the derivative of the Wiener process. As the Wiener process does not have a well-defined derivative, the precise interpretation of the white noise process depends on the way the derivative is performed. The most widely used interpretations are those of Itô and Stratonovich. To illustrate this, let us consider the integral

$$\int_t^{t'} f(x(s))\xi(s)ds \tag{6.42}$$

where $f(x(t))$ is an arbitrary function of the stochastic process $x(t)$ whose dynamics is given by a Langevin equation of the form

$$\frac{dx}{dt} = q(x) + g(x)\xi(t). \tag{6.43}$$

The integral (6.42) is sometimes also written as

$$\int_t^{t'} f(x(s))dW \tag{6.44}$$

although here we will mainly use the notation $\xi(s)ds$ rather than dW. As the integral depends on a stochastic process, each realization of the stochastic process will lead to a different value for the integral, thus integrals of this form are called *stochastic integrals*.

Now, we first proceed to compute the integral (6.42) for $t' = t + h$ in the limit $h \to 0$. In the so-called Itô interpretation, the result of the integral is

$$\int_t^{t+h} f(x(s))\xi(s)ds = f(x(t))[W(t+h) - W(t)] \tag{6.45}$$

namely the function is evaluated at the initial time. In the so-called Stratonovich interpretation, the result of the integral is

$$\int_t^{t+h} f(x(s))\xi(s)ds = f\left(\frac{x(t)+x(t+h)}{2}\right)[W(t+h)-W(t)]. \tag{6.46}$$

That is, the function is evaluated at a point which is the average of the value of $x(t)$ at the initial and final times. In some sense, the existence of the different definitions is related to the definition of the following expression:

$$\int_0^{\infty} dt\ \delta(t) \tag{6.47}$$

which is equal to 1 (Itô) or to 1/2 (Stratonovich).

For a finite integration time, we divide the integration interval $[t, t']$ into N subintervals and take the limit $N \to \infty$. In Itô calculus, this leads to

$$\int_t^{t'} f(x(s))\xi(s)ds = \lim_{N\to\infty}\sum_{i=1}^{N} f(x(t_{i-1}))[W(t_i)-W(t_{i-1})] \tag{6.48}$$

where $t_i = t + ih$ with $h = (t'-t)/N$. In Stratonovich calculus, the integral is given by

$$\int_t^{t'} f(x(s))\xi(s)ds = \lim_{N\to\infty}\sum_{i=1}^{N} f\left(\frac{x(t_{i-1})+x(t_i)}{2}\right)[W(t_i)-W(t_{i-1})]. \tag{6.49}$$

In general, one could have evaluated the stochastic integral as

$$\int_t^{t'} f(x(s))\xi(s)ds = \lim_{N\to\infty}\sum_{i=1}^{N} f\left(\alpha x(t_{i-1})+(1-\alpha)x(t_i)\right)[W(t_i)-W(t_{i-1})] \tag{6.50}$$

with any arbitrary α such that $0 \le \alpha \le 1$. The Ito interpretation corresponds to $\alpha = 1$, whereas the Stratonovich interpretation corresponds to $\alpha = 1/2$.[4)]

In the past, there has been much argument on which was the "correct" interpretation. There is no such thing. It is just a matter of convention. This means that a Langevin stochastic differential equation with white noise (6.43) is not completely defined unless we specify the interpretation considered. Once the interpretation has been fixed, then any expression such as (6.42) has a unique meaning and its result can be uniquely determined.

In many cases of interest, the Stratonovich interpretation turns out to be more "natural" since it commutes with the limit given by (6.29) and (6.30). In this book, unless otherwise stated, we will always use the Stratonovich interpretation. Besides, the Stratonovich interpretation allows the use of the familiar rules of calculus, such as the change of variables in an integration step. Instead, in the Itô interpretation, in some instances, one cannot use the ordinary rules of calculus and instead use specific ones, which tend to be more cumbersome. One advantage of the Itô interpretation is that the stochastic process at time t, $x(t)$, is not correlated with the

4) The choice $\alpha = 0$ corresponds to Klimontovich interpretation of the stochastic integral.

white noise acting on the system at the same time $\xi(t)$: that is, $\langle x(t)\xi(t)\rangle = 0$. This is not the case with the Stratonovich interpretation.

To illustrate the differences between the Itô and Stratonovich interpretations, let us consider the simple integral

$$\int_{t_a}^{t_b} W(s)\xi(s)ds\,. \tag{6.51}$$

In Itô calculus, this leads to

$$\int_{t_a}^{t_b} W(s)\xi(s)ds = \lim_{N\to\infty}\sum_{i=1}^{N} W(t_{i-1})\left[W(t_i) - W(t_{i-1})\right] =$$

$$= \frac{1}{2}\left[W(t_b)^2 - W(t_a)^2\right] - \frac{1}{2}\lim_{N\to\infty}\sum_{i=1}^{N}\left[W(t_i) - W(t_{i-1})\right]^2. \tag{6.52}$$

In Stratonovich calculus, this leads to

$$\int_{t_a}^{t_b} W(s)\xi(s)ds = \lim_{N\to\infty}\sum_{i=1}^{N}\frac{1}{2}\left[W(t_{i-1}) + W(t_i)\right]\left[W(t_i) - W(t_{i-1})\right] =$$

$$= \lim_{N\to\infty}\frac{1}{2}\sum_{i=1}^{N}\left[W(t_i)^2 - W(t_{i-1})^2\right] = \frac{1}{2}\left[W(t_b)^2 - W(t_a)^2\right]\,. \tag{6.53}$$

Thus, while Stratonovich calculus leads to a result for the integral which is what one would expect from ordinary calculus, in the Itô interpretation there is a nonintuitive additional term. Technically, the limit is to be understood as the mean square limit. A series F_n converges to F in mean square if

$$\lim_{n\to\infty}\langle(F_n - F)^2\rangle = 0. \tag{6.54}$$

It can be shown, (see Exercise 3) that the mean square limit of $\sum_{i=1}^{N}\left[W(t_i) - W(t_{i-1})\right]^2$ is $t_b - t_a$. Therefore, in Itô calculus

$$\int_{t_a}^{t_b} W(s)\xi(s)ds = \frac{1}{2}\left[W(t_b)^2 - W(t_a)^2 - (t_b - t_a)\right]\,. \tag{6.55}$$

We should also mention that there is a simple relation between the Langevin stochastic differential equations written in both interpretations. The rule is that the Langevin equation

$$\frac{dx}{dt} = q_{\rm I}(x) + g_{\rm I}(x)\xi(t) \tag{6.56}$$

in the Itô sense is equivalent to the Langevin equation

$$\frac{dx}{dt} = q_{\rm I}(x) - \frac{1}{2}g_{\rm I}(x)g_{\rm I}'(x) + g_{\rm I}(x)\xi(t) \tag{6.57}$$

in the Stratonovich sense. Therefore, the equivalent Stratonovich equation to a given Itô equation has an additional drift term. Notice that, when the noise is additive, $g(x)$ is a constant, and then the Langevin equation is the same in both interpretations.

6.6 The Ornstein–Uhlenbeck Process

The white noise we have introduced in Section 6.4 is in fact a mathematical idealization that allows some simplifications and also obtaining some analytical results. For instance, it can be shown that the solution $x(t)$ of a Langevin stochastic differential equation (6.24) is a Markov process if the noise is white. However, random fluctuations affecting physical, chemical, biological or other systems have typically a nonzero correlation time. For example, in the Brownian process, there is a timescale given by the collision time of the water molecules with the Brownian particle beyond which fluctuations cannot be considered uncorrelated. It just happens that this timescale is much smaller than the typical timescales in which the Brownian particle moves, and therefore the white noise approximation is justified when one is interested in describing the ''slow'' movement of the ''big'' Brownian particle. However, there are instances in which the separation between the timescales is not very large and then one must consider a Langevin stochastic differential equation in which the noise term does not have a zero correlation time. An example is the emission in a dye laser, in which the hydrodynamic fluctuations in the flow of the dye are at a timescale which is not faster than the characteristic scales for stimulated emission.

A stochastic process widely used to model fluctuations with a finite correlation time is the Ornstein–Uhlenbeck noise $\xi^{\mathrm{ou}}(t)$. It is formally defined as a Gaussian Markov process characterized by

$$\langle \xi^{\mathrm{ou}}(t) \rangle = 0, \tag{6.58}$$

$$\langle \xi^{\mathrm{ou}}(t)\xi^{\mathrm{ou}}(s) \rangle = \frac{1}{2\tau} \mathrm{e}^{-|t-s|/\tau} \tag{6.59}$$

that is, the correlations of the noise decay exponentially in time with a characteristic timescale given by the parameter τ. In the limit $\tau \to 0$, the exponential becomes a Dirac delta function and the Ornstein–Uhlenbeck process becomes a white noise process. We note that, when performing this limit, the resulting stochastic differential equation should be interpreted in the Stratonovich sense. It can also be shown that, up to a change of variables, the Ornstein–Uhlenbeck process is the only Gaussian, Markov, stationary process.

The Ornstein–Uhlenbeck process can be seen as the solution of the Langevin stochastic differential equation

$$\frac{d\xi^{\mathrm{ou}}(t)}{dt} = -\frac{1}{\tau}\xi^{\mathrm{ou}}(t) + \frac{1}{\tau}\xi(t) \tag{6.60}$$

where $\xi(t)$ is a white noise defined by (6.39) and (6.40), and taking as initial condition that $\xi^{\mathrm{ou}}(0)$ is a Gaussian random variable of mean 0 and variance $(2\tau)^{-1}$. Furthermore, for an arbitrary initial condition, the stationary solution of (6.60) is also an Ornstein–Uhlenbeck process (see Exercise 1).

6.6.1 Colored Noise

The Ornstein–Uhlenbeck process is an example of "colored noise." By this notation, one usually refers to stochastic processes acting on Langevin equations for which the correlation time is not a Dirac delta function. The name comes as opposition to white noise which has a flat spectrum. Instead, noises that have a finite correlation time have a nonflat spectrum. We also mention that, although in principle "colored noise" could refer to any noise with finite correlation time, in many instances it is used as a synonym of an Ornstein–Uhlenbeck process.

6.7 The Fokker–Planck Equation

So far, we have focused on trajectories to describe stochastic processes following the initial Langevin approach for Brownian motion. The alternative approach introduced by Einstein focuses on probabilities rather than in trajectories and, as it happens with the Langevin approach, it can be extended way beyond the Brownian motion. In what follows, we are going to determine an equation for one-time probability distribution for a stochastic process described by a Langevin equation with white noise. This is called the *Fokker–Planck equation* and it is a generalization of the diffusion equation obtained by Einstein to describe the Brownian process.

To be more precise, we want to find an equation for the one-time probability distribution function $f(x,t)$ for a stochastic process $x(t)$ which arises as a solution of a stochastic differential equation with a Gaussian white noise $\xi(t)$ defined in (6.39) and (6.40)

$$\frac{dx}{dt} = q(x) + g(x)\xi(t). \tag{6.61}$$

This is to be understood in the Stratonovich interpretation. Let us consider first the corresponding deterministic equation

$$\frac{dx}{dt} = q(x), \qquad x(t=0) = x_0. \tag{6.62}$$

The solution of this equation is a (deterministic) function $x(t) = F(t, x_0)$. Nevertheless, for our purposes, we can see $x(t)$ as a random variable whose probability density function $\rho(x;t)$ gives the probability to find the system at a given location x in phase space at a time t. As the trajectory followed by the system is uniquely determined once the initial condition is given, this probability is zero everywhere except at the location $x = F(t, x_0)$. Therefore, the probability density function is a delta function, given by

$$\rho(x;t) = \delta(x - F(t, x_0)). \tag{6.63}$$

We can now transform $x(t)$ into a stochastic process by simply letting the initial condition x_0 to become a random variable. We have now an ensemble of trajectories,

each one starting from a different initial condition x_0. In this case, the probability density function $\rho(x,t)$ is obtained by averaging the above pdf over the distribution of the initial conditions:

$$\rho(x;t) = \langle \delta(x - F(t,x_0)) \rangle_{x_0}. \tag{6.64}$$

In fact, $\rho(x;t)$ can be seen as a density function. To visualize it, we assume that at time $t = 0$ we have unleashed a large number of particles N with one particle at each of the initial conditions we are considering in our ensemble. We let the particles evolve according to the dynamics and, after a time t, $\rho(x;t)dx$ measures the fraction of particles located in the interval $(x, x+dx)$:

$$\rho(x;t)dx = \frac{\#\text{particles in } (x,x+dx) \text{ at time } t}{\text{total \# of particles}} = \frac{n(x,t)}{N}dx. \tag{6.65}$$

Assuming there are no sinks nor sources of particles, then the number of particles at a given location changes as a result of the number of particles crossing the borders. Let $J(x,t)$ be the flux of particles:

$$J(x,t)dt = \frac{\#\text{ particles that cross the point } x \text{ in the interval } (t,t+dt)}{N}. \tag{6.66}$$

J has a direction. For convenience, we consider $J > 0$ if the particle moves to the right. Then

$$\rho(x;t+dt)dx - \rho(x;t)dx = J(x,t)dt - J(x+dx,t)dt. \tag{6.67}$$

The first term on the left-hand side (LHS) is the final number of particles at x, whereas the second is the initial one. Thus, the LHS corresponds to the variation of the number of particles at x from time t to time $t+dt$. On the right-hand side (RHS), the first term is the number of particles that have entered or left during the interval $(t, t+dt)$ through the left boundary, whereas the second term measures the number of particles that crossed the right boundary. Therefore, one has

$$\frac{\rho(x;t+dt) - \rho(x;t)}{dt} = \frac{J(x,t) - J(x+dx,t)}{dx} \tag{6.68}$$

which in the continuous limit corresponds to

$$\frac{\partial \rho(x;t)}{\partial t} = -\frac{\partial J(x,t)}{\partial x} \tag{6.69}$$

which is the continuity equation.

As particles move following a deterministic dynamics given by (6.62), trajectories cannot cross and therefore in a one-dimensional system they cannot advance from one to the other. The particles that will cross the point x in the interval $(t, t+dt)$ are all those located in the interval $dx = q(x)dt$. Therefore,

$$J(x,t) = \frac{n(x,t)}{N}\frac{dx}{dt} = \rho(x;t)q(x). \tag{6.70}$$

Replacing this in (6.69), one gets the Liouville equation

$$\frac{\partial \rho(x;t)}{\partial t} = -\frac{\partial}{\partial x}[q(x)\rho(x;t)]. \tag{6.71}$$

We consider now the full stochastic differential equation (6.61). We can repeat the above argument for a given realization of the noise term. The probability density function $f(x;t)$ will be the average of $\rho(x;t)$ with respect to the noise distribution:

$$f(x;t) = \langle \rho(x;t) \rangle_\xi = \langle \delta(x - x(t, x_0)) \rangle_{x_0,\xi}. \tag{6.72}$$

$\rho(x;t)$ satisfies the continuity equation (6.69). Now the current is given by

$$J(x,t) = \rho(x;t)\frac{dx}{dt} = \rho(x;t)\left[q(x) + g(x)\xi(t)\right]. \tag{6.73}$$

Therefore,

$$\frac{\partial \rho(x;t)}{\partial t} = -\frac{\partial}{\partial x}[(q(x) + g(x)\xi(t))\rho]. \tag{6.74}$$

By taking averages over the noise term, we get

$$\begin{aligned}\frac{\partial f(x;t)}{\partial t} &= \left\langle \frac{\partial \rho(x;t)}{\partial t} \right\rangle_\xi = -\frac{\partial}{\partial x}\left\langle \left[q(x) + g(x)\xi(t)\right] \rho(x;t) \right\rangle_\xi \\ &= -\frac{\partial}{\partial x}\left[q(x)\langle \rho(x;t)\rangle_\xi\right] - \frac{\partial}{\partial x}\left[g(x)\langle \xi(t)\rho(x;t)\rangle_\xi\right].\end{aligned} \tag{6.75}$$

The averages on the first term of the RHS can be easily performed $\langle \rho(x;t)\rangle_\xi = f(x;t)$. However, averages on the second term of the RHS are more cumbersome, since one has to keep in mind that the density distribution ρ depends on $x(t)$, which itself depends functionally on the noise through the dynamics (6.61). In fact, to be precise, the average on the second term of the RHS should be written as $\langle \xi(t)\rho(x[\xi(t)];t)\rangle_\xi$. This average can be obtained by using Novikov's theorem[5], which establishes that, for any Gaussian stochastic process $\xi_G(t)$ with zero mean, $\langle \xi_G(t)\rangle = 0$, and for any functional of the noise $\mathcal{F}[\xi_G(t)]$ one has

$$\langle \xi_G(t)\mathcal{F}[\xi_G(t)]\rangle_{\xi_G} = \int_0^t ds \langle \xi_G(t)\xi_G(s)\rangle_{\xi_G} \left\langle \frac{\delta \mathcal{F}}{\delta \xi_G(s)} \right\rangle_{\xi_G} \tag{6.76}$$

where the last term is the functional derivative of $\mathcal{F}[\xi_G(t)]$ with respect to $\xi_G(s)$. We note that Novikov theorem is quite general; it only requires the noise to be Gaussian but does not require it to be white, namely, the noise correlations need not be a delta function.

In our case, we have a Gaussian white noise, for which the correlation is a delta function. This allows us to evaluate the integral easily:

$$\begin{aligned}\langle \xi(t)\rho(x[\xi(t)];t)\rangle_\xi &= \int_0^t ds\ \delta(t-s)\left\langle \frac{\delta \rho(x[\xi(t)];t)}{\delta \xi(s)} \right\rangle_\xi \\ &= \frac{1}{2}\left\langle \frac{\delta \rho(x[\xi(t)];t)}{\delta \xi(s)}\bigg|_{s=t} \right\rangle_\xi.\end{aligned} \tag{6.77}$$

5) We use the extension to a Gaussian process of the theorem whose proof was proposed in Exercise 1.10 for a finite number of Gaussian variables.

By using the chain rule in functional calculus, this can be computed as follows:

$$\left\langle \frac{\delta\rho(x[\xi(t)];t)}{\delta\xi(s)}\bigg|_{s=t} \right\rangle_\xi = \left\langle \frac{\delta x(t)}{\delta\xi(s)}\bigg|_{s=t} \frac{\partial\rho(x;t)}{\partial x(t)} \right\rangle_\xi$$
$$= -\frac{\partial}{\partial x}\left\langle \frac{\delta x(t)}{\delta\xi(s)}\bigg|_{s=t} \rho(x;t) \right\rangle_\xi . \tag{6.78}$$

To evaluate the functional derivative of the stochastic process $x(t)$ with respect to the noise, we use a formal solution of (6.61):

$$x(t) = x_0 + \int_0^t ds\, q(x(s)) + \int_0^t ds\, g(x(s))\xi(s). \tag{6.79}$$

Then

$$\frac{\delta x(t)}{\delta\xi(s)}\bigg|_{s=t} = g(x(t)), \tag{6.80}$$

and

$$\left\langle \frac{\delta\rho(x[\xi(t)];t)}{\delta\xi(s)}\bigg|_{s=t} \right\rangle_\xi = -\frac{\partial}{\partial x}[g(x(t))\langle\rho(x;t)\rangle_\xi] = -\frac{\partial}{\partial x}[g(x)f(x;t)]. \tag{6.81}$$

Using this result in (6.75) we get, finally, the Fokker–Planck equation for the probability density function:

$$\frac{\partial f(x;t)}{\partial t} = -\frac{\partial}{\partial x}\left[q(x)f(x;t)\right] + \frac{1}{2}\frac{\partial}{\partial x}\left[g(x)\frac{\partial}{\partial x}\left[g(x)f(x;t)\right]\right]. \tag{6.82}$$

The procedure we have used to derive the Fokker–Planck equation for a single variable can be extended to the case of several variables. Consider a set of stochastic variables $\vec{x} = (x_1, x_2, \dots, x_N)$ whose dynamics is given by the set of Langevin equations to be considered in the Stratonovich interpretation:

$$\frac{dx_i}{dt} = q_i(\vec{x},t) + \sum_{j=1}^{N} g_{ij}(\vec{x},t)\xi_j(t), \quad i = 1 \dots N \tag{6.83}$$

where we allow for the drift terms $q_i(\vec{x},t)$ and diffusion terms $g_{ij}(\vec{x},t)$ to explicitly depend on the time. $\xi_j(t)$ are uncorrelated Gaussian white noises with zero mean, that is

$$\langle\xi_i(t)\rangle = 0 \quad \langle\xi_i(t)\xi_j(s)\rangle = \delta_{i,j}\delta(t-s). \tag{6.84}$$

By using a straightforward extension of the method used for one variable, one can prove that the one-time probability density function $f(\vec{x};t)$ satisfies the following multivariate Fokker–Planck equation:

$$\frac{\partial f(\vec{x};t)}{\partial t} = -\sum_{i=1}^{N}\frac{\partial}{\partial x_i}\left[q_i(\vec{x},t)f(\vec{x};t)\right]$$
$$+ \frac{1}{2}\sum_{i,j,k=1}^{N}\frac{\partial}{\partial x_i}\left[g_{ik}(\vec{x},t)\frac{\partial}{\partial x_j}\left[g_{jk}(\vec{x},t)f(\vec{x};t)\right]\right]. \tag{6.85}$$

It can also be written in the form of a continuity equation

$$\frac{\partial f(\vec{x};t)}{\partial t} + \sum_{i=1}^{N} \frac{\partial}{\partial x_i} J_i(\vec{x},t) = 0 \tag{6.86}$$

where the probability currents $J_i(\vec{x},t)$ are given by

$$J_i(\vec{x},t) = q_i(\vec{x},t) f(\vec{x};t) - \frac{1}{2} \sum_{j,k=1}^{N} g_{ik}(\vec{x},t) \frac{\partial}{\partial x_j} \left[g_{jk}(\vec{x},t) f(\vec{x};t) \right]. \tag{6.87}$$

If $q_i(\vec{x},t)$ and $g_{ij}(\vec{x},t)$ do not depend explicitly on time, namely $q_i(\vec{x},t) = q_i(\vec{x})$ and $g_{ij}(\vec{x})$, then the Fokker–Planck is called *homogeneous*.

6.7.1 Stationary Solution

In general, solutions of the Fokker–Planck equation are difficult to be found analytically. An exception is the stationary solution in the case of a single variable. In this case, imposing

$$\left. \frac{\partial f(x;t)}{\partial t} \right|_{f(x,t)=f^{\mathrm{st}}(x)} = 0 \tag{6.88}$$

one has that in the stationary state the current must fulfill

$$\frac{\partial J^{\mathrm{st}}(x)}{\partial x} = 0. \tag{6.89}$$

That is, $J^{\mathrm{st}}(x)$ must be a constant. The simplest situation is when this constant is zero, which means that in the stationary state there is no flux of probability, and then one has

$$q(x) f^{\mathrm{st}}(x) - \frac{g(x)}{2} \frac{d}{dx} \left[g(x) f^{\mathrm{st}}(x) \right] = 0. \tag{6.90}$$

This can be rewritten as

$$\frac{2q(x) - g(x)g'(x)}{g^2(x)} dx = \frac{df^{\mathrm{st}}(x)}{f^{\mathrm{st}}(x)}. \tag{6.91}$$

Integrating on both sides, and considering that a and b are the lower and upper boundaries for the variable x, one has

$$\int_a^x d\gamma \frac{2q(\gamma) - g(\gamma)g'(\gamma)}{g^2(\gamma)} = \log[f^{\mathrm{st}}(\gamma)]\Big|_a^x \tag{6.92}$$

Then

$$f^{\mathrm{st}}(x) = C\Psi(x) \tag{6.93}$$

where

$$\Psi(x) = \exp \int_a^x d\gamma \frac{2q(\gamma) - g(\gamma)g'(\gamma)}{g^2(\gamma)} = \frac{|g(a)|}{|g(x)|} \exp \int_a^x d\gamma \frac{2q(\gamma)}{g^2(\gamma)} \tag{6.94}$$

and C is a normalization constant such that

$$\int_a^b f^{\mathrm{st}}(x) = 1. \tag{6.95}$$

We now consider the case in which the stationary current is a nonzero constant $J^{\mathrm{st}}(x) = J$. This situation, in which there is a stationary flux of probability, is usually encountered in systems with periodic boundary conditions, for which $f^{\mathrm{st}}(a) = f^{\mathrm{st}}(b)$, with a and b being the left and right boundaries of the system, respectively. J is not arbitrary; its value is determined by the normalization of $f^{\mathrm{st}}(x)$ and by the boundary value of the probability density function. In this case, one has

$$q(x)f^{\mathrm{st}}(x) - \frac{g(x)}{2}\frac{d}{dx}\left[g(x)f^{\mathrm{st}}(x)\right] = J \tag{6.96}$$

which is an equation similar to (6.89) but with an additional inhomogeneous term. It can be shown (see Exercise 4) that the current is given by

$$J = \left[\frac{g(b)^2}{\Psi(b)} - \frac{g(a)^2}{\Psi(a)}\right]\frac{f^{\mathrm{st}}(a)}{\int_a^b \frac{dy}{\Psi(y)}} \tag{6.97}$$

and the stationary solution is

$$f^{\mathrm{st}}(x) = C\frac{\frac{g^2(b)}{\Psi(b)}\int_a^x \frac{dy}{\Psi(y)} + \frac{g^2(a)}{\Psi(a)}\int_x^b \frac{dy}{\Psi(y)}}{\frac{g^2(x)}{\Psi(x)}\int_a^b \frac{dy}{\Psi(y)}} \tag{6.98}$$

where C is a normalization constant.

Further Reading and References

For a more detailed historical explanation of the Brownian motion, see, for example, the books by Risken [17], Pathria [18], or Gardiner [19]. The early work on Boltzmann on Brownian motion was published in [20]. Einstein's description of this phenomenon was published in 1905 [21]. Langevin's approach was published 3 years later [22, 23].

For more information on stochastic processes, see, for example, the books by Gardiner [19], Van Kampen [24], and Weiss [25]. For early works on stochastic processes, see [26]. For an extensive study of the Fokker–Planck equation, see, for instance, the book by Risken [17]. For a review of the effects of noise in physical systems, see the review by San Miguel and Toral [27].

Itô calculus is named after Kiyoshi Itô, who introduced a method to evaluate the stochastic integral. For a historical review of his work, see [28]. For an introduction to Itô calculus, see, for instance, the book by Gardiner [19].

The Stratonovich stochastic integral was developed simultaneously by Ruslan Leont'evich Stratonovich and Donald L. Fisk as an alternative to Itô calculus which allows the application of ordinary rules of calculus to stochastic differential equations. For a review on Stratonovich calculus, see the book by Stratonovich, translated to English by Silverman [29].

The Ornstein–Uhlenbeck process is named after Leonard Salomon Ornstein and George Eugene Uhlenbeck who introduced it to describe the velocity of a Brownian particle with inertia [30].

Exercises

1) Consider the linear stochastic differential equation (6.60).
   a. Solve the equation. Hint, solve as a nonhomogeneous first-order linear differential equation in which $\xi(t)$ can be treated as an arbitrary function of the time.
   b. Starting from an arbitrary initial condition, evaluate $\langle \xi^{\mathrm{ou}}(t) \rangle$ and $\langle \xi^{\mathrm{ou}}(t)\xi^{\mathrm{ou}}(s) \rangle$ and show that, for $t \to \infty$, (6.58) and (6.59) hold.
   c. Consider as initial condition a Gaussian random number of zero mean and variance $1/2\tau$. Show that in this case (6.58) and (6.59) hold at any time.
2) Consider the stochastic differential equation $\dot{x}(t) = -ax + \sqrt{D}\xi^{\mathrm{ou}}(t)$, where $\xi^{\mathrm{ou}}(t)$ is a Gaussian colored noise of zero mean and correlation $\langle \xi^{\mathrm{ou}}(t)\xi^{\mathrm{ou}}(t') \rangle = (\frac{1}{2\tau})\mathrm{e}^{-|t-t'|/\tau}$. Take as initial condition $x(t=0) = x_0$.
   a. Solve the equation, that is, write $x(t)$ as a function of the noise.
   b. Calculate $\langle x(t) \rangle$. Discuss the existence of a transient regime and a stationary regime. Show that in the stationary regime $\langle x(t) \rangle = 0$.
   c. Calculate $C(t,s) = \langle x(t)x(t+s) \rangle$. Show that in the stationary regime $C(t,s)$ is only a function of s.
3) Consider the stochastic integral

$$\int_{t_a}^{t_b} W(s)\xi(s)ds$$

which in the Itô calculus is evaluated as

$$\lim_{N\to\infty} \sum_{i=1}^{N} W(t_{i-1}) \left[W(t_i) - W(t_{i-1}) \right]$$

where $t_i = t_a + ih$ with $h = (t_b - t_a)/N$.
   a. Show that this last sum is equivalent to

$$\frac{1}{2}\left[W(t_b)^2 - W(t_a)^2 \right] - \frac{1}{2} \lim_{N\to\infty} \sum_{i=1}^{N} \left[W(t_i) - W(t_{i-1}) \right]^2 .$$

   b. Show that

$$\left\langle \sum_{i=1}^{N} [W(t_i) - W(t_{i-1})]^2 \right\rangle = t_b - t_a .$$

   c. Show that

$$\left\langle \left[\sum_{i=1}^{N} \left[W(t_i) - W(t_{i-1}) \right]^2 - (t_b - t_a) \right]^2 \right\rangle = 2 \sum_{i=1}^{N} (t_i - t_{i-1})^2 .$$

d. Show that the value of this last expression tends to zero in limit $N \to \infty$.

4) Show that the stationary solution of the Fokker–Planck equation with periodic boundary conditions and nonzero current is given by (6.98). To do so, proceed as follows:
   a. It is useful to use the function $\Psi(x)$ given by (6.94). By differentiating $\Psi(x)$, show that
   $$2q(x) = \frac{g(x)^2\Psi'(x)}{\Psi(x)} - g(x)g'(x).$$
   b. Substitute this result into (6.96) and show that
   $$\frac{d}{dx}\frac{g(x)^2 f^{\mathrm{st}}(x)}{\Psi(x)} = \frac{2J}{\Psi(x)}.$$
   c. Integrate both sides. Use the periodic boundary condition $f^{\mathrm{st}}(a) = f^{\mathrm{st}}(b)$ to show that the current is given by (6.97).
   d. Using the result of the integration and the value for the current, show that the stationary solution is given by (6.98). What is the value of C?

5) Consider the stochastic differential equation $\dot{x}(t) = ax - x^3 + \sqrt{D}\xi(t)$, where $\xi(t)$ is a Gaussian white noise with zero mean and correlation $\langle \xi(t)\xi(t')\rangle = \delta(t - t')$. This equation describes a supercritical pitchfork bifurcation with noise.
   a. Write the corresponding Fokker–Planck equation.
   b. Find the stationary distribution. Discuss how the shape of the stationary distribution changes with a.
   c. Find the values of x at which the stationary distribution has a maximum or a minimum as function of a.

6) Consider the stochastic differential equation $\dot{z}(t) = (\mu + i\omega)z - |z|^2 z + \sqrt{D}\xi(t)$, where z is a complex variable and $\xi(t)$ is a complex Gaussian white noise with zero mean and correlation $\langle \xi(t)\xi^*(t')\rangle = 2\delta(t - t')$. This equation describes a supercritical Hopf bifurcation with noise.
   a. Write the equation for the real and imaginary parts $z = z_R + \mathrm{i}z_I$ and $\xi = \xi_R + \mathrm{i}\xi_I$. Show that the correlation for the complex white noise stated earlier is compatible with $\left\langle \xi_R(t)\xi_R(t')\right\rangle = \left\langle \xi_I(t)\xi_I(t')\right\rangle = \delta(t - t')$ and $\left\langle \xi_R(t)\xi_I(t')\right\rangle = 0$.
   b. Write the corresponding Fokker–Planck equation.
   c. Find the stationary distribution. Discuss how the shape of the stationary distribution changes with μ.

7) Given the stochastic differential equation (Stratonovich sense)
   $$\dot{x} = -\tanh(x) + \mathrm{sech}(x)\sqrt{2D}\xi(t)$$
   where $\xi(t)$ is the white noise of zero mean and correlations $\langle \xi(t)\xi(t')\rangle = \delta(t - t')$, and $x \in (-\infty, +\infty)$ is a real variable.
   a. Write down the Fokker–Planck equation.
   b. Under the condition of zero current, find the stationary probability distribution $P_{\mathrm{st}}(x)$.

c. Write it in the form $P_{st}(x) = \mathcal{Z}^{-1}e^{-V(x)/D}$ and find the effective potential $V(x)$.
d. Find the critical value of the noise intensity D_c for which the effective potential changes from monostable to bistable.

8) Consider the equation $\dot{x}(t) = ax - bx^3 + \sqrt{D}x\xi(t)$, where $\xi(t)$ is a Gaussian white noise defined in (6.39) and (6.40).
a. Prove that the dynamics does not allow crossing at $x = 0$. The system then possesses a natural barrier at this location.
b. Write down the Fokker–Planck equation and find the stationary solution depending on the initial condition.

9) Prove that the stationary solution of the multivariate Fokker–Planck equation (6.83) in the particular case of a drift, that is

$$f_i = -\sum_{j=1}^{N} S_{ij}\frac{\partial V}{\partial x_j}$$

where $V(x_1, \dots, x_N)$ is the ''potential'' function ad S is a constant positive-definite symmetric matrix related to the matrix of diffusion coefficients as $S = 2\epsilon gg^{\intercal}$, is

$$f^{st}(x_1, \dots, x_N) = Ce^{-V/\epsilon}.$$

10) Show that the Fokker-Planck equation associated to the Ito Langevin equation $\dot{x} = Q(x) + \sqrt{G(x)}\xi(t)$ is given by

$$\frac{\partial f(x;t)}{\partial t} = \frac{\partial}{\partial x}\left(-Q(x)f(x;t) + \frac{1}{2}\frac{\partial}{\partial n}\left(G(x)f(x;t)\right)\right)$$

This result will be used at the end of Chapter 8.

7
Numerical Simulation of Stochastic Differential Equations

In Chapter 6, we provided a basic overview on stochastic processes and introduced two complementary descriptions: the first one, in terms of stochastic differential equations, comes from Langevin's treatment of the Brownian motion that focuses on trajectories; the second one, in terms of the Fokker–Planck equation, focuses on probabilities as in Einstein's treatment of the Brownian motion. In principle, both ways allow the characterization of the stochastic process, such as calculating averages or correlations. However, in many instances, looking at the individual trajectories allows one to extract valuable information that is not so easy to obtain from the time evolution of the probability density function. In fact, the generation and visualization of some representative trajectories is usually quite helpful in providing physical understanding of what is going on, as illustrated in Section 7.6.

Therefore, here we focus on numerical methods to generate trajectories. To be more precise, consider that we have a given Langevin stochastic differential equation and we would like to obtain with a computer several representative trajectories. As the noise will be different in each trajectory, even if we start always from the same initial condition, all the trajectories will be different and, in principle, there will be an infinite number of them. Therefore, we cannot aim at generating all possible trajectories, but just a finite number of them. Still, if properly done, this finite number of trajectories can be sufficient to obtain averages or correlations with a certain degree of accuracy.

Because for a given realization of the noise a stochastic equation becomes an ordinary differential equation, one may naively consider that it can be numerically integrated using any standard method, such as the Euler method, the popular fourth-order Runge–Kutta method, or a predictor–corrector method. Therefore, one may question the need for this chapter altogether. The point is that these standard methods cannot be used to integrate typical stochastic differential equations. To illustrate this, let us consider the equation

$$\dot{x}(t) = q(x) + g(x)\xi(t) \tag{7.1}$$

where, as usual, $\xi(t)$ is a Gaussian white noise of mean and correlations given by (6.39) and (6.40).

The standard methods to integrate the ordinary differential equations $\dot{x}(t) = q(x)$ assume that the functions $q(x)$ are well behaved to a certain extent. For example,

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

the Euler method assumes that these functions are differentiable. Runge–Kutta methods or predictor–corrector methods assume that they are differentiable to a higher order. However, as described in Chapter 6, the white noise is not a differentiable function. Even for a single realization of the white noise term, $\xi(t)$ is highly irregular and not differentiable even at first order. As discussed in that chapter, it can be thought of as a series of Dirac delta functions spread all over the real axis. In fact, the numerical value that the noise takes at a given time is not properly defined since it is a Gaussian random number with infinite variance. As a consequence, we cannot directly use the standard methods for ordinary differential equations. If one were dealing with a stochastic differential equation with smooth functions as random processes, we could certainly use the standard methods. This is not the usual case: the stochastic contribution is a non-analytical function and it is necessary to use a different kind of algorithms. The basic idea behind the construction of these new algorithms is to integrate: whereas the values of the white noise or its derivatives are not well defined, the integrals are (for instance, the first integral is the Wiener process, which is a continuous function).

7.1 Numerical Integration of Stochastic Differential Equations with Gaussian White Noise

Let us start with a simple stochastic differential equation

$$\dot{x}(t) = f(t) + \xi(t)\,. \tag{7.2}$$

If fact, as the right-hand side does not depend on x, this equation can be solved exactly, the solution being

$$x(t) = x(0) + \int_0^t f(s)ds + \int_0^t \xi(s)ds \tag{7.3}$$

which can be written as

$$x(t) = x(0) + F(t) + W(t) \tag{7.4}$$

where $F(t) \equiv \int_0^t f(s)ds$, and $W(t)$ is the Wiener process. This expression indicates that $x(t)$ is a Gaussian process whose mean and correlation are given by

$$\begin{aligned} \langle x(t)\rangle &= x(0) + F(t), \\ \langle x(t)x(t')\rangle &= [x(0) + F(t)]\left[x(0) + F(t')\right] + \min(t, t'). \end{aligned} \tag{7.5}$$

In what follows, we disregard that the equation can be solved and instead focus on a numerical solution in which we generate trajectories: that is, we want to obtain $x(t)$ at discrete time intervals

$$x(t+h) = x(t) + \int_t^{t+h} \dot{x}(s)ds = x(t) + \int_t^{t+h} f(s)ds + \int_t^{t+h} \xi(s)ds. \tag{7.6}$$

Introducing

$$f_h(t) \equiv \int_t^{t+h} f(s)ds, \tag{7.7}$$

and

$$w_h(t) \equiv \int_t^{t+h} \xi(s)ds = W(t+h) - W(t) \tag{7.8}$$

we can write

$$x(t+h) = x(t) + f_h(t) + w_h(t). \tag{7.9}$$

In this equation, as $w_h(t)$ is the difference of the Wiener process at two different times, it is a Gaussian process and can be fully characterized by giving the mean and correlations:

$$\langle w_h(t)\rangle = \int_t^{t+h} \langle \xi(s)\rangle ds = 0, \tag{7.10}$$

$$\begin{aligned}\langle w_h(t)w_h(t')\rangle &= \int_t^{t+h}\int_{t'}^{t'+h} \langle \xi(s)\xi(u)\rangle dsdu \\ &= \int_t^{t+h}\int_{t'}^{t'+h} \delta(s-u)dsdu.\end{aligned} \tag{7.11}$$

To evaluate the integral, we make use of the properties of the Dirac δ function. We can assume, without loss of generality, that $t' > t$. If $t' > t+h$, the integral is 0, as there is no overlap in the integration intervals, and the delta function vanishes. If $t \le t' < t+h$, the double integral equals the length of the overlap Interval, that is

$$\langle w_h(t)w_h(t')\rangle = \int_{t'}^{t+h} ds = t - t' + h. \tag{7.12}$$

In particular, for $t' = t$, one has

$$\langle w_h(t)^2\rangle = h. \tag{7.13}$$

In numerical calculations, time takes always discrete values as multiples of the integration time step. If we consider discrete times $t = t_i = ih$, $t' = t_j = jh$, the correlation becomes

$$\langle w_h(t_i)w_h(t_j)\rangle = h\delta_{ij}. \tag{7.14}$$

We introduce now a set of independent Gaussian random variables $\{u_i\}$ of zero mean and variance 1:

$$\langle u_i\rangle = 0, \qquad \langle u_i u_j\rangle = \delta_{ij} \tag{7.15}$$

in terms of which we can write

$$w_h(t_i) = \sqrt{h}u_i. \tag{7.16}$$

The set of independent Gaussian variables $\{u_i\}$ can be generated using the methods discussed in Chapter 3. Finally, the recurrence relation to generate numerically trajectories of the stochastic process defined by (7.2) is

$$\begin{aligned}x(t_0) &= x_0, \\ x(t_{i+1}) &= x(t_i) + f_h(t_i) + \sqrt{Dh}u_i.\end{aligned} \tag{7.17}$$

For small h, the deterministic contribution $f_h(t)$ can be approximated as $f_h(t) = hf(t)$, implying that for small h the deterministic contribution is of order h^1 and successive contributions go as h^2, h^3, and so on. In contrast the stochastic contribution is of order $h^{1/2}$, an order that never appears in methods for numerical integration of ordinary differential equations. A consequence of this scaling with the time step is that at very small times the stochastic contribution, being of order $h^{1/2}$, dominates over the deterministic one, of order h. Over the course of an integration for a large period, the stochastic contributions on average will be generally of less importance than the deterministic ones.

The recurrence equation (7.17) can be readily implemented in a program that allows the generation of a numerical trajectory. Averages and correlations can be obtained by integrating many trajectories with independent values for the noise. Besides, if the initial condition is given by a probability distribution, it is also necessary to sample this distribution of initial conditions.

We now apply the same ideas to a stochastic differential equation of the general form given by (7.1). We discretize the time $t = t_i = t_0 + ih$, where t_0 is the time of the initial condition, $i = 0, 1, 2, \ldots$, and h is the integration time step. Our goal is to obtain a recurrence relation that provides the value of $x(t_{i+1})$ as a function of $x(t_i)$. Proceeding as before, we write

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} q(x(s))ds + \int_{t_i}^{t_{i+1}} g(x(s))\xi(s)ds. \tag{7.18}$$

Now we assume that the functions $q(x(s))$ and $g(x(s))$ are differentiable functions and expand them in a Taylor series around $x = x(t_i)$:

$$q(x(s)) = q(x(t_i)) + \left.\frac{dq(x)}{dx}\right|_{x(t_i)} \left(x(s) - x(t_i)\right) + O\left[\left(x(s) - x(t_i)\right)^2\right], \tag{7.19}$$

$$g(x(s)) = g(x(t_i)) + \left.\frac{dg(x)}{dx}\right|_{x(t_i)} \left(x(s) - x(t_i)\right) + O\left[\left(x(s) - x(t_i)\right)^2\right]. \tag{7.20}$$

Substitution of these expansions into (7.18) leads to

$$\begin{aligned} x(t_{i+1}) = x(t_i) &+ hq(x(t_i)) + q'(x(t_i)) \int_{t_i}^{t_{i+1}} \left(x(s) - x(t_i)\right) ds + hO\left[\left(x(s) - x(t_i)\right)^2\right] \\ &+ w_h(t_i)g(x(t_i)) + g'(x(t_i)) \int_{t_i}^{t_{i+1}} \left(x(s) - x(t_i)\right) \xi(s)ds \\ &+ w_h(t_i)O\left[\left(x(s) - x(t_i)\right)^2\right] \end{aligned} \tag{7.21}$$

where we used the notation

$$q'(x(t_i)) \equiv \left.\frac{dq(x)}{dx}\right|_{x(t_i)}, \quad g'(x(t_i)) \equiv \left.\frac{dg(x)}{dx}\right|_{x(t_i)}. \tag{7.22}$$

As in the simple previous example, $w_h(t_i)$ is given by (7.16) and is of order $h^{1/2}$. In (7.21), we can identify the deterministic contribution (terms involving $q(x, t)$) and the noise contribution (terms involving $g(x, t)$). Now, we want to keep all the terms up to order h. On the deterministic part, the third (the integral) and the

fourth terms of the first line are of higher order and can be disregarded. On the stochastic part, the term $w_h(t_i)g(x(t))$ is only of order $h^{1/2}$, so we may need to keep also some other terms. To do so, we need to evaluate the integral present in the second term of the stochastic part, namely

$$\int_{t_i}^{t_{i+1}} \big(x(s) - x(t_i)\big)\, \xi(s) ds. \tag{7.23}$$

The evaluation of this integral requires the knowledge of $x(s)$ at a time s between t_i and t_{i+1} which is not known *a priori* (in fact, if we knew it for all s, there will be no need for of the numerical algorithm we are developing here). As we do not know $x(s)$, a way to proceed is to estimate $x(s)$ using an approximation. To zero order, $x(s) = x(t_i)$, which does not give any contribution in the integral (7.23). The next order is $h^{1/2}$, and to this order one has

$$x(s) - x(t_i) = g(x(t_i)) \int_{t_i}^{s} \xi(u) du + O[h]. \tag{7.24}$$

Introducing (7.24) into the integral (7.23), we have

$$\int_{t_i}^{t_{i+1}} \big(x(s) - x(t_i)\big)\, \xi(s) ds = g(x(t_i)) \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} \xi(u)\xi(s) ds du + \omega_h(t) O[h]. \tag{7.25}$$

Since $\omega_h(t)$ is of order $h^{1/2}$, the last term is of order $h^{3/2}$. The double integral can be readily evaluated as

$$\begin{aligned} \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} \xi(u)\xi(s) ds du &= \int_{t_i}^{t_{i+1}} \big[W(s) - W(t_i)\big]\, \xi(s) ds \\ &= \int_{t_i}^{t_{i+1}} W(s)\xi(s) ds - W(t_i) \int_{t_i}^{t_{i+1}} \xi(s) ds. \end{aligned} \tag{7.26}$$

As discussed in Section 6.5 in Stratonovich calculus, the first integral is given by $[W(t_{i+1})^2 - W(t_i)^2]/2$. Therefore,

$$\begin{aligned} \int_{t_i}^{t_{i+1}} \int_{t_i}^{s} \xi(u)\xi(s) ds du &= \frac{W(t_{i+1})^2 - W(t_i)^2}{2} - W(t_i)\big[W(t_{i+1}) - W(t_i)\big] \\ &= \frac{1}{2}\big[W(t_{i+1})^2 - 2W(t_{i+1})W(t_i) + W(t_i)^2\big] \\ &= \frac{1}{2}\big[W(t_{i+1}) - W(t_i)\big]^2 = \frac{w_h(t_i)^2}{2}. \end{aligned} \tag{7.27}$$

Therefore, the second term of the stochastic part of the right-hand side of (7.21) is given by

$$g'(x(t_i)) \int_{t_i}^{t_{i+1}} \big(x(s) - x(t_i)\big)\, \xi(s) ds = \frac{1}{2} g'(x(t_i)) g(x(t_i)) w_h(t_i)^2 + O[h^{3/2}]. \tag{7.28}$$

We note that the corrections given by the last term on the right-hand side of (7.21) are of higher order:

$$w_h(t_i) O\Big[\big(x(s) - x(t_i)\big)^2\Big] = O[h^{3/2}] \tag{7.29}$$

and therefore can be disregarded.

Collecting all the results, up to order h, (7.21) can be written as

$$x(t_{i+1}) = x(t_i) + hq(x(t_i)) + g(x(t_i))h^{1/2}u_i + \frac{1}{2}g(x(t_i))g'(x(t_i))hu_i^2 + O[h^{3/2}]. \quad (7.30)$$

This recurrence relation is known in the literature as the *Milshtein algorithm*.

There is an alternative way to obtain this result starting from (7.18). According to Stratonovich calculus,

$$\int_{t_i}^{t_{i+1}} g(x(s))\xi(s)ds = g\left(\frac{x(t_{i+1}) + x(t_i)}{2}\right) w_h(t_i)\,. \quad (7.31)$$

Using the Taylor expansion (7.20)

$$g\left(\frac{x(t_{i+1}) + x(t_i)}{2}\right) = g(x(t_i)) + g'(x(t_i))\frac{x(t_{i+1}) - x(t_i)}{2} + O\left[\left(x(t_{i+1}) - x(t_i)\right)^2\right]\,. \quad (7.32)$$

As $w_h(t_i)$ is of order $h^{1/2}$, in (7.32) we only need to evaluate $x(t_{i+1}) - x(t_i)$ at the lowest order, namely $h^{1/2}$, which is given by

$$\begin{aligned} x(t_{i+1}) - x(t_i) &= g(x(t_i)) \int_{t_i}^{t_{i+1}} \xi(u)du + O[h] \\ &= g(x(t_i))w_h(t_i) + O[h]\,. \end{aligned} \quad (7.33)$$

Substituting this into (7.32) and further the result into (7.31) gives

$$\int_{t_i}^{t_{i+1}} g(x(s))\xi(s)ds = g(x(t_i))w_h(t_i) + \frac{1}{2}g(x(t_i))g'(x(t_i))w_h(t_i)^2 + O[h^{3/2}] \quad (7.34)$$

which combined with

$$\int_{t_i}^{t_{i+1}} q(x(s))ds = hq(x(t_i)) + O[h^{3/2}] \quad (7.35)$$

leads to the Milshtein algorithm (7.30).

It is possible to perform an additional approximation to the Milshtein algorithm in which the u_i^2 term is replaced by its mean value $\langle u_i^2\rangle = 1$, leading to an algorithm of the same order:

$$x(t_{i+1}) = x(t_i) + hq(x(t_i)) + g(x(t_i))h^{1/2}u_i + \tfrac{1}{2}g(x(t_i))g'(x(t_i))h + O[h^{3/2}]. \quad (7.36)$$

Although the two algorithms are of the same order, for a single stochastic differential equation, this second approximation is unnecessary because it is very easy to evaluate u_i^2. Therefore (7.30) is preferred over (7.36).

If the noise is additive, $g(x(t))$ does not depends on x, and $g(x(t)) = \sqrt{D}$ with D being a constant, then the resulting algorithm is called the *Euler* or *Euler–Maruyama algorithm*

$$x(t+h) = x(t) + hq(x(t)) + \sqrt{Dh}u(t) + O[h^{3/2}]\,. \quad (7.37)$$

While the Euler–Maruyama algorithm is suitable for additive noise only, it is sometimes used for stochastic differential equations with multiplicative noise:

$$x(t_{i+1}) = x(t_i) + hq(x(t_i)) + g(x(t_i))h^{1/2}u_i + O[h]\,. \tag{7.38}$$

Notice that the error at each time step is of order $O[h]$, which is pretty bad. Therefore, we strongly advise against using the Euler–Maruyama algorithm for stochastic differential equations with multiplicative noise.

We note, however, that, while the Euler–Maruyama is not suitable for stochastic differential equations understood in the Stratonovich sense, it is appropriate for Itô stochastic differential equations: that is, if one considers

$$\dot{x} = q_{\mathrm{I}}(x) + g_{\mathrm{I}}(x)\xi(t) \tag{7.39}$$

then Euler–Maruyama algorithm reads

$$x(t_{i+1}) = x(t_i) + hq_{\mathrm{I}}(x(t_i)) + g_{\mathrm{I}}(x(t_i))h^{1/2}u_i + O[h^{3/2}]\,. \tag{7.40}$$

In fact, by using (6.57), (7.40) turns out to be equivalent to (7.36). For an Itô stochastic differential equation, the equivalent Milshtein algorithm (7.30) is written as

$$\begin{aligned} x(t_{i+1}) = x(t_i) + hq_{\mathrm{I}}(x(t_i)) + g_{\mathrm{I}}(x(t_i))h^{1/2}u_i + \frac{1}{2}g_{\mathrm{I}}(x(t_i))g_{\mathrm{I}}'(x(t_i))h(u_i^2 - 1) \\ + O[h^{3/2}]. \end{aligned} \tag{7.41}$$

Note that the simpler algorithm (7.40) has the same order of convergence.

Coming back to Stratonovich calculus, we would like to note that, in general, the Milshtein algorithm can be readily used in stochastic differential equations in which the drift or the diffusion terms depend explicitly on the time: namely

$$\dot{x}(t) = q(t, x) + g(t, x)\xi(t). \tag{7.42}$$

In this case, the Milshtein algorithm is given by the map

$$\begin{aligned} x(t_{i+1}) = x(t_i) + hq(t_i, x(t_i)) + g(t_i, x(t_i))h^{1/2}u_i \\ + \frac{1}{2}g(t_i, x(t_i))g'(t_i, x(t_i))hu_i^2 + O[h^{3/2}] \end{aligned} \tag{7.43}$$

where, now, since g is a function of two variables, by $g'(t, x)$ we mean $\partial g/\partial x$.

7.1.1 Integration Error

In the previous expressions, we wrote that the correction to the recurrence algorithms is of order $O[h^{3/2}]$. We now explain in more detail what we mean by this because, in fact, for a stochastic process, there are several ways to estimate the order of the algorithm. Consider that we know the value of the stochastic process at time t, $x(t)$. We apply a single step of the Milshtein algorithm to estimate the value of the stochastic process at time $t_{i+1} = t_i + h$:

$$\tilde{x}(t_{i+1}) = x(t_i) + hq(t_i, x(t_i)) + g(t_i, x(t_i))h^{1/2}u_i + \frac{1}{2}g(t_i, x(t_i))g'(t_i, x(t_i))hu_i^2 \tag{7.44}$$

where we have used the notation $\tilde{x}$ for the estimation of the value of the stochastic process given by the Milshtein algorithm. For each realization of the noise, we compare the estimation $\tilde{x}(t_{i+1})$ with the exact value $x(t_{i+1})$ and evaluate the square error. Then, it can be shown that the average over realizations of the squared error $R(h)$ is of order $O[h^3]$:

$$R(h) \equiv \langle (\tilde{x}(t+h) - x(t+h))^2 \rangle = O[h^3]. \tag{7.45}$$

Technically, one says that the Milshtein algorithm has a convergence for the trajectories in the mean square of order h^3.

There is another way of estimating the order of the algorithm and looking at the convergence of the moments. The order of convergence of the nth order moment, $D_n(h)$, is defined as

$$D_n(h) \equiv \langle \tilde{x}(t+h)^n \rangle - \langle x(t)^n \rangle \tag{7.46}$$

starting from a given $x(t)$ and performing the averages over noise realizations. For the Milshtein algorithm, it can be shown that

$$D_n(h) = O[h^2]. \tag{7.47}$$

This means that when computing moments, the error in each integration time step is of order $O[h^2]$. Typically, one starts from t_0 and wants to compute the value of the stochastic process after a finite time $t_f - t_0 = Nh$; therefore, it is legitimate to ask for an estimation of the error at the end of the integration. This depends on how the error propagates, which, in turn, depends on the dynamics of the system. If the deterministic dynamics of the system is not chaotic, one may assume that the errors propagate in a linear way. Under this assumption, the total error after the N integration steps is of order $NO[h^2] = ((t_f - t_0)/h)O[h^2] = O[h]$. In other words, after a finite time, the error in the evaluation of the moments is of order $O[h]$:

$$\langle x(t_f)^n \rangle = \langle \tilde{x}(t_f)^n \rangle + O[h]. \tag{7.48}$$

In practice, what one should do is to repeat the numerical integration using several time steps $h_1 > h_2 > h_3 > \cdots$ and extrapolate the results to $h = 0$.

When integrating ordinary differential equations, one typically uses algorithms in which the error is of much higher order than $O[h^{3/2}]$, such as, for example, a fourth-order Runge–Kutta algorithm, where the error at each time step is of order $O[h^5]$. Therefore, it is natural to ask for a higher order method than (7.30) for stochastic processes, something of the form

$$\begin{aligned} x(t+h) = x(t) &+ F_1(x)h + F_2(x)w_h(t) + F_3(x)w_h(t)^2 \\ &+ F_4(x)hw_h(t) + F_5(x)w_h(t)^3 + \dots . \end{aligned} \tag{7.49}$$

In principle, one can try to extend the previous calculations and obtain such development. However, for a stochastic equation of the general form (7.1), as soon as one goes beyond the Milshtein method, the development involves more random processes, which are not Gaussian and which have nontrivial correlations among them. As a consequence, in general, it is very difficult to generate these terms

appropriately and the resulting algorithms are more cumbersome. Therefore, in practice, the simplest way to integrate a stochastic differential equation is to use the Milshtein algorithm (7.30). In Section 7.4, we will introduce a Runge–Kutta-type method for stochastic differential equations which for white noise has an error in the trajectories of order $O[h^{3/2}]$ as the Milshtein algorithm.

We now provide a routine to generate a trajectory of a stochastic differential equation using the Milshtein algorithm:

```
program Milshtein
implicit none
double precision :: x,t,x_0,t_0,h,h_sqrt,uh
double precision :: q,g,gprime,ran_g
integer :: i_step,n_step,i_write,n_write
external q,g,gprime

x_0=1.d0
t_0=0.d0
h=0.01d0
n_write =200
n_step=10

h_sqrt=sqrt(h)

x=x_0
t=t_0
do i_write=1,n_write
  do i_step=1,n_step
    uh=h_sqrt*ran_g()
    x=x+uh*g(t,x)+h*q(t,x)+0.5d0*g(t,x)*gprime(t,x)*uh**2
    t=t+h
  enddo
  write (20,*) t,x
enddo

end program Milshtein
```

The core of the program is a double loop in which the Milshtein algorithm is applied repetitively. Typically, one wants to store the values of the trajectory at time intervals that are much larger than the time step used for integration. Thus, the internal loop performs `n_step` integration steps between writing data to a file. The program integrates the trajectory up to a final time which is `n_step*n_write*h` writing data every `n_step*h` time units. The functions `q(t,x)`, `g(t,x)`, and `gprime(t,x)` should be provided externally. As it is written, the function `g(t,x)` is called twice with the same arguments at each time step. To avoid that, the central line of the inner loop can be rewritten as

```
x=x+h*q(t,x)+uh*g(t,x)*(1.d0+0.5d0*gprime(t,x)*uh).
```

If one is going to evaluate averages, then an additional loop over trajectories is needed. In this case, one can first generate many trajectories, store them in

a file (or in several files), and then use an additional program to compute the averages. Alternatively (and more efficiently), one may calculate the averages along with the numerical integration, reducing the need for storage. To illustrate this last procedure, consider that you want to evaluate the first two moments of the stochastic variable every `n_step*h` time units; a simple program to do that would be as follows:

```
program Milshtein_averages
implicit none
double precision :: x,t,x_0,t_0,h,h_sqrt,uh
double precision, dimension (:), allocatable :: xm_1,xm_2
double precision :: q,g,gprime,ran_g
integer :: i_step,n_step,i_write,n_write,i_tra,n_tra
external q,g,gprime

x_0=1.d0
t_0=0.d0
h=0.01d0
n_tra=100
n_write=200
n_step=10

allocate (xm_1(n_write))
allocate (xm_2(n_write))
xm_1=0.d0
xm_2=0.d0

h_sqrt=sqrt(h)

do i_tra=1,n_tra
  x=x_0
  t=t_0
  do i_write=1,n_write
    do i_step=1,n_step
     uh=h_sqrt*ran_g()
     x=x+h*q(t,x)+uh*g(t,x)*(1.d0+0.5d0*gprime(t,x)*uh)
     t=t+h
    enddo
    xm_1(i_write)=xm_1(i_write)+x
    xm_2(i_write)=xm_2(i_write)+x**2
  enddo
enddo

xm_1=xm_1/n_tra
xm_2=xm_2/n_tra
do i_write=1,n_write
  write (20,*) i_write*n_step*h,xm_1(i_write),xm_2(i_write)
enddo

end program Milshtein_averages
```

7.2
The Ornstein–Uhlenbeck Process: Exact Generation of Trajectories

As explained in Section 6.6, the Ornstein–Uhlenbeck stochastic process can be seen as the stationary solution of the Langevin stochastic differential equation (6.60), which is driven by a Gaussian white noise. One could integrate that equation using the Milshtein algorithm described in Section 7.1, but this requires the use of very small time steps. There is a more efficient approach. Equation (6.60) is linear and, therefore, it can be solved exactly over one time step, allowing for larger time steps. The solution is

$$\xi^{\mathrm{ou}}(t_{i+1}) = \xi^{\mathrm{ou}}(t_i)\mathrm{e}^{-h/\tau} + H_h(t_i) \tag{7.50}$$

where

$$H_h(t) = \frac{\mathrm{e}^{-(t+h)/\tau}}{\tau}\int_t^{t+h} \xi(s)\mathrm{e}^{s/\tau}ds \tag{7.51}$$

is a Gaussian stochastic process and therefore is fully characterized by its mean and correlation. The mean value of $H_h(t)$ can be readily evaluated from its definition

$$\langle H_h(t)\rangle = \frac{\mathrm{e}^{-(t+h)/\tau}}{\tau}\int_t^{t+h} \langle\xi(s)\rangle\mathrm{e}^{s/\tau}ds = 0. \tag{7.52}$$

The correlations are given by

$$\begin{aligned}\langle H_h(t)H_h(t')\rangle &= \frac{\mathrm{e}^{-(t+t'+2h)/\tau}}{\tau^2}\int_t^{t+h}\int_{t'}^{t'+h}\langle\xi(s)\xi(u)\rangle\mathrm{e}^{(s+u)/\tau}dsdu \\ &= \frac{\mathrm{e}^{-(t+t'+2h)/\tau}}{\tau^2}\int_t^{t+h}\int_{t'}^{t'+h}\mathrm{e}^{(s+u)/\tau}\delta(s-u)dsdu.\end{aligned} \tag{7.53}$$

The double integral can be evaluated in a way similar to that of (7.11). Assume that $t' > t$. If $t' > t + h$, the integral is 0 as there is no overlap in the integration intervals and the delta function vanishes. If $t \le t' < t + h$, then

$$\langle H_h(t)H_h(t')\rangle = \frac{\mathrm{e}^{-(t+t'+2h)/\tau}}{\tau^2}\int_{t'}^{t+h}\mathrm{e}^{2s/\tau}ds = \frac{\mathrm{e}^{(t-t')/\tau}}{2\tau}\left[1-\mathrm{e}^{-2h/\tau}\right]. \tag{7.54}$$

In general, for any arbitrary t and t', the correlation is nonzero only for $|t - t'| < h$, in which case it is given by

$$\langle H_h(t)H_h(t')\rangle = \frac{\mathrm{e}^{-|t-t'|/\tau}}{2\tau}\left[1-\mathrm{e}^{-2h/\tau}\right]. \tag{7.55}$$

For the discrete times $t_i = t_0 + ih$ that appear in the map given by (6.60), the correlations are

$$\langle H_h(t_i)H_h(t_j)\rangle = \frac{1}{2\tau}\left[1-\mathrm{e}^{-2h/\tau}\right]\delta_{ij}. \tag{7.56}$$

To generate the random numbers $H_h(t_i)$, we start from a set of independent Gaussian random variables $\{u_i\}$ of zero mean and variance 1 as given by (7.15). In

terms of them, we can write

$$H_h(t_i) = \sqrt{\frac{1 - e^{-2h/\tau}}{2\tau}}\, u_i. \tag{7.57}$$

Finally, we obtain a recurrence relation to generate numerically realizations of the Ornstein–Uhlenbeck noise:

$$\xi^{ou}(t_0) = \frac{1}{\sqrt{2\tau}} u_0,$$
$$\xi^{ou}(t_{i+1}) = \xi^{ou}(t_i) e^{-h/\tau} + \sqrt{\frac{1 - e^{-2h/\tau}}{2\tau}}\, u_{i+1}. \tag{7.58}$$

We would like to remark that (7.58) is exact for any arbitrary time step h. Below, we include a program that generates a trajectory of the Ornstein–Uhlenbeck process in which the time step is as large as the time interval at which we write the results.

```
program OU_process
implicit none
double precision :: xiOU,tau,h,p,coeff,ran_g
integer :: i_write, n_write

h=0.2d0
tau=0.5d0
n_write=200

p=exp(-h/tau)
coeff=sqrt((1.d0-exp(-2.d0*h/tau)/(2.d0*tau))

 xiOU=ran_g()/sqrt(2.d0*tau)
 write (20,*) 0.d0,xiOU
 do i_write=1,n_write
   xiOU=xiOU*p+coeff*ran_g()
   write (20,*) i_write*h,xiOU
 enddo

end program OU_process
```

7.3 Numerical Integration of Stochastic Differential Equations with Ornstein–Uhlenbeck Noise

In this section, we address the generation of trajectories for a stochastic differential equation driven by an Ornstein–Uhlenbeck noise. As in Section 7.1, we start with the simplest example

$$\dot{x}(t) = f(t) + \xi^{ou}(t). \tag{7.59}$$

Also, as before, we use an integral algorithm:

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} f(s)ds + \int_{t_i}^{t_{i+1}} \xi^{ou}(s)ds \equiv x(t_i) + f_h(t_i) + g_h(t_i). \tag{7.60}$$

The stochastic contribution $g_h(t)$ is a Gaussian process characterized by its mean and correlation values. The mean can be readily calculated from the definition

$$\langle g_h(t)\rangle = \int_{t_i}^{t_{i+1}} \langle \xi^{\mathrm{ou}}(s)\rangle ds = 0. \tag{7.61}$$

For the correlation, we have

$$\begin{aligned}\langle g_h(t_i) g_h(t_j)\rangle &= \int_{t_i}^{t_{i+1}} \int_{t_j}^{t_{j+1}} \langle \xi^{\mathrm{ou}}(s)\xi^{\mathrm{ou}}(u)\rangle ds du \\ &= \frac{1}{2\tau}\int_{t_i}^{t_{i+1}} \int_{t_j}^{t_{j+1}} \mathrm{e}^{-|s-u|/\tau} ds du.\end{aligned} \tag{7.62}$$

For $t_i = t_j$, the double integral can be evaluated as follows:

$$\begin{aligned}&\frac{1}{2\tau}\int_{t_i}^{t_{i+1}} \left[\int_{t_i}^{s} \mathrm{e}^{-(s-u)/\tau} du + \int_{s}^{t_{i+1}} \mathrm{e}^{(s-u)/\tau} du\right] ds \\ &= \frac{1}{2}\int_{t_i}^{t_{i+1}} \left[2 - \mathrm{e}^{(t_i-s)/\tau} - \mathrm{e}^{(s-t_{i+1})/\tau}\right] ds = h + \tau\left(\mathrm{e}^{-h/\tau} - 1\right).\end{aligned} \tag{7.63}$$

For $t_i \neq t_j$, there is no overlap in the domain of integration of the two integrals. Assume $t_i < t_j$, then the double integral can be written as

$$\begin{aligned}\frac{1}{2\tau}\int_{t_i}^{t_{i+1}} \int_{t_j}^{t_{j+1}} \mathrm{e}^{(s-u)/\tau} ds\, du &= \frac{1-\mathrm{e}^{-h/\tau}}{2}\int_{t_i}^{t_{i+1}} \mathrm{e}^{(s-t_j)/\tau} \\ &= \tau\left[\frac{\mathrm{e}^{h/\tau}+\mathrm{e}^{-h/\tau}}{2} - 1\right]\mathrm{e}^{(t_i-t_j)/\tau}.\end{aligned} \tag{7.64}$$

Therefore, for $t_i \neq t_j$, one has

$$\langle g_h(t_i) g_h(t_j)\rangle = \tau\left[\cosh\left(\frac{h}{\tau}\right) - 1\right]\mathrm{e}^{-|t_i-t_j|/\tau}. \tag{7.65}$$

One can show that in the limit $\tau \to 0$, g_h becomes a white noise, more precisely $g_h(t_i) \to \sqrt{h}u_i$.

To begin with, both $g_h(t_i)$ and $\sqrt{h}u_i$ are Gaussian stochastic processes with zero mean. So we only need to check the behavior of the correlation. For equal times, from (7.63) we have

$$\lim_{\tau\to 0}\langle g_h(t_i)^2\rangle = \lim_{\tau\to 0} h + \tau\left(\mathrm{e}^{-h/\tau} - 1\right) = h. \tag{7.66}$$

Thus, in the limit $\tau \to 0$, the second moment of $g_h(t_i)$ is the same as $\sqrt{h}u_i$. For $t_i \neq t_j$ and small τ, the correlation of g_h given by (7.65) becomes

$$\begin{aligned}\lim_{\tau\to 0}\langle g_h(t_i) g_h(t_j)\rangle &= \lim_{\tau\to 0}\frac{\tau}{2}\mathrm{e}^{h/\tau}\mathrm{e}^{-|t_i-t_j|/\tau} \\ &= \lim_{\tau\to 0}\frac{\tau}{2}\mathrm{e}^{(1-|i-j|)h/\tau} = 0.\end{aligned} \tag{7.67}$$

For $i = j \pm 1$, the limit goes to zero linearly with τ. If i and j differ by more than 1, then the limit decays to zero exponentially. In any case, in the limit $\tau \to 0$, the

stochastic process $g_h(t)$ becomes uncorrelated at different times. Collecting all the results, it is clear that for $\tau \to 0$ the process $g_h(t)$ goes continuously to $\sqrt{h}u_i$.

As a separate matter, for small integration time step h, it can be seen that, up to order $O[h^2]$, both the correlations at equal times (7.63) and different (7.65) times can be written as

$$\langle g_h(t_i)g_h(t_j)\rangle = \frac{h^2}{2\tau}e^{|t_i-t_j|/\tau} + O[h^3]. \tag{7.68}$$

Comparing this result with the correlation for an Ornstein–Uhlenbeck process given by (6.59), it is clear that at order $O[h^2]$ the process $g_h(t)$ can be approximated by an Ornstein–Uhlenbeck process multiplied by h, that is

$$g_h(t) \approx h\xi^{ou}(t). \tag{7.69}$$

The advantage of this approximation is that $\xi^{\text{ou}}(t)$ can be easily generated as explained earlier. However, we will see in the next subsection that the process $g_h(t)$ can be generated without resorting to any approximation.

Summarizing, the algorithm to integrate numerically the stochastic differential equation (7.59) is

$$x(t_{i+1}) = x(t_i) + f_h(t_i) + g_h(t_i) + O[h^2] \tag{7.70}$$

where g_h can be generated exactly as explained in 7.3.1 or approximated by (7.69) with $\xi^{\text{ou}}(t_i)$ generated as indicated in (7.58).

We now consider the numerical integration of a more general stochastic differential equation with an Ornstein–Uhlenbeck noise:

$$\dot{x}(t) = q(t,x) + g(t,x)\xi^{\text{ou}}(t). \tag{7.71}$$

As before, we discretize the time and integrate formally both sides of the equation to obtain a map:

$$x(t_{i+1}) = x(t_i) + \int_{t_i}^{t_{i+1}} q(s,x(s))ds + \int_{t_i}^{t_{i+1}} g(s,x(s))\xi^{\text{ou}}(s)ds. \tag{7.72}$$

By Taylor-expanding the functions $q(s,x(s))$ and $g(s,x(s))$, one can verify that at the lowest order

$$x(t_{i+1}) = x(t_i) + hq(t_i,x(t_i)) + g_h(t_i)g(t_i,x(t_i)) + O[h^2]. \tag{7.73}$$

As before, we can use the approximation (7.69) for $g_h(t)$ and then generate $\xi^{\text{ou}}(t_i)$ as indicated in (7.58) or use the exact generation described in subsection 7.3.1.

The algorithms (7.70) and (7.73) suffer from the fact that they do not reproduce adequately the white noise of the Milshtein method when taking the limit $\tau \to 0$. In fact, the integration time step h has to be kept smaller than the correlation time τ. If one is interested in the limit of small values for the correlation time, then it is necessary to keep reducing the integration time step, which makes these algorithms impracticable for this purpose. In the next section, in the context of the Runge–Kutta-type methods, we will revisit this discussion.

7.3.1
Exact Generation of the Process $g_h(t)$

This sub-section is devoted to explaining a way to generate exactly the process $g_h(t)$. At this point, this may be considered a sort of technical issue of little practical relevance since using the approximation (7.69) in (7.73) does not affect the order of the integration method. However, we will see in the next section that the exact generation of the process $g_h(t)$ is indeed useful in some instances. We define the process

$$G(t) = \int_{t_0}^{t} ds\, \xi^{\mathrm{ou}}(s). \tag{7.74}$$

In terms of the above, $g_h(t)$ can be written as

$$g_h(t_i) = G(t_{i+1}) - G(t_i). \tag{7.75}$$

As $dG(t)/dt = \xi^{\mathrm{ou}}(t)$ satisfies the differential equation (6.60), $G(t)$ obeys the following stochastic differential equation with white noise:

$$\frac{d^2G(t)}{dt^2} + \frac{1}{\tau}\frac{dG(t)}{dt} = \frac{1}{\tau}\xi(t) \tag{7.76}$$

with the initial conditions

$$G(t_0) = 0, \qquad \left.\frac{dG(t)}{dt}\right|_{t=t_0} = \xi^{\mathrm{ou}}(t_0) \equiv \xi_0 \tag{7.77}$$

where $\xi^{\mathrm{ou}}(t_0)$ is a Gaussian random number with zero mean and variance $\langle \xi^{\mathrm{ou}}(t_0)^2 \rangle = 1/2\tau$. The solution of (7.76) is then given by

$$G(t) = \tau\xi_0 \left(1 - \mathrm{e}^{-(t-t_0)/\tau}\right) + \int_{t_0}^{t} ds\, \xi(s) - \mathrm{e}^{-(t-t_0)/\tau} \int_{t_0}^{t} ds\, \mathrm{e}^{s/\tau}\xi(s). \tag{7.78}$$

From (7.75) and (7.78), we can obtain a recurrence relation for g_h:

$$g_h(t_{i+1}) = pg_h(t_i) - pf_1(t_i) + f_1(t_{i+1}) - f_2(t_i) + f_2(t_{i+1}) \tag{7.79}$$

where

$$p = \mathrm{e}^{-h/\tau}, \tag{7.80}$$

$$f_1(t) = \int_{t}^{t+h} ds\, \xi(s), \tag{7.81}$$

$$f_2(t) = -p\mathrm{e}^{-t/\tau} \int_{t}^{t+h} ds\, \mathrm{e}^{s/\tau}\xi(s). \tag{7.82}$$

The processes $f_1(t)$ and $f_2(t)$ are correlated Gaussian processes, whose properties, for the discrete times $t_i = ih$ appearing in the recurrence relations, are given by

$$\begin{aligned}
&\langle f_1(t_i)\rangle = \langle f_2(t_i)\rangle = 0,\\
&\langle f_1(t_i) f_1(t_j)\rangle = h\delta_{ij},\\
&\langle f_2(t_i) f_2(t_j)\rangle = \frac{\tau(1-p^2)}{2}\delta_{ij},\\
&\langle f_1(t_i) f_2(t_j)\rangle = -\tau(1-p)\delta_{ij}.
\end{aligned} \tag{7.83}$$

Therefore, f_1 and f_2 are Gaussian variables uncorrelated at different times, but correlated at equal times. As explained in Section 3.5, at each time, they can be generated from two independent Gaussian random variables u_i and v_i as follows:

$$\begin{aligned} f_1(t_i) &= \alpha u_i, \\ f_2(t_i) &= \beta u_i + \gamma v_i \end{aligned} \tag{7.84}$$

where the constants α, β, and γ are chosen appropriately to fulfill the correlations (7.83):

$$\alpha = \sqrt{h}, \tag{7.85}$$

$$\beta = -\frac{\tau(1-p)}{\sqrt{h}}, \tag{7.86}$$

$$\gamma = \sqrt{\frac{\tau(1-p)}{2}\left[1 - \frac{2\tau}{h} + p\left(1 + \frac{2\tau}{h}\right)\right]}. \tag{7.87}$$

So, finally, the algorithm to generate exactly the process $g_h(t)$ is as follows: In the initial time step, we set

$$\begin{aligned} f_1(t_0) &= \alpha u_0, \\ f_2(t_0) &= \beta u_0 + \gamma v_0, \\ g_h(t_0) &= \sqrt{\frac{\tau}{2}}(1-p)u_g + f_1(t_0) + f_2(t_0) \end{aligned} \tag{7.88}$$

where u_g is a Gaussian random number of zero mean and variance 1 independent of $\{u_i\}$ and $\{v_i\}$. Note that the initial value for $g_h(0)$ is set so that its second moment fulfills (7.63):

$$\begin{aligned} \langle g_h(t_0)^2 \rangle &= \frac{\tau}{2}(1-p)^2\langle u_g^2 \rangle + \langle f_1(t_0)^2 \rangle + \langle f_2(t_0)^2 \rangle + 2\langle f_1(t_0) f_2(t_0) \rangle \\ &= \frac{\tau}{2}(1-p)^2 + h + \frac{\tau(1-p^2)}{2} - 2\tau(1-p) = h + \tau(p-1). \end{aligned} \tag{7.89}$$

Then for the other time steps we use the recurrence

$$\begin{aligned} f_1(t_{i+1}) &= \alpha u_{i+1}, \\ f_2(t_{i+1}) &= \beta u_{i+1} + \gamma v_{i+1}, \\ g_h(t_{i+1}) &= p g_h(t_i) - p f_1(t_i) + f_1(t_{i+1}) - f_2(t_i) + f_2(t_{i+1}). \end{aligned} \tag{7.90}$$

An alternative way to initialize this recurrence is as follows: We formally define

$$\begin{aligned} f_1(t_{-1}) &= 0, \\ f_2(t_{-1}) &= -\sqrt{\frac{\tau}{2}}(1-p)u_{-1}, \\ g_h(t_{-1}) &= 0 \end{aligned} \tag{7.91}$$

where u_{-1} is a Gaussian random number of zero mean and variance 1. Then, applying the general expression for the recurrence (7.90) one gets for $t = t_0$, $g_h(t_0)$ defined as in (7.88). We will make use of this computational trick in what follows.

We provide now an numerical implementation of the function gh together with a subroutine gh_init that calculates the parameters and initializes the process using the computational trick (7.91).

```
subroutine gh_init(tau,h)
 implicit none
 double precision :: tau,h
 double precision :: h_sqrt,p,beta,gamma,f1_old,f2_old,gh_aux,ran_g
 common /gh_data/ h_sqrt,p,beta,gamma,f1_old,f2_old,gh_aux
 h_sqrt=sqrt(h)
 p=exp(-h/tau)
 beta=-tau*(1.d0-p)/h_sqrt
 gamma= sqrt(tau*(1.d0-p)*(1.d0-2.d0*tau/h+p*(1+2.d0*tau/h))/2.d0)
 f1_old=0.d0
 f2_old=-sqrt(tau/2.d0)*(1.d0-p)*ran_g()
 gh_aux=0.d0
end subroutine gh_init


double precision function gh()
 implicit none
 double precision :: h_sqrt,p,beta,gamma,f1_old,f2_old,gh_aux
 double precision :: u,v,f1,f2,ran_g
 common /gh_data/ h_sqrt,p,beta,gamma,f1_old,f2_old,gh_aux
  u=ran_g()
  v=ran_g()
  f1=h_sqrt*u
  f2=beta*u+gamma*v
  gh_aux=p*(gh_aux-f1_old)+f1-f2_old+f2
  f1_old=f1
  f2_old=f2
  gh=gh_aux
end function gh
```

These routines can be used for correlation time τ smaller than the integration time step h. We will take advantage of this in conjunction with the Heun method discussed in the next section.

The case $\tau = 0$ requires some further discussion since the evaluation of $p = e^{-h/\tau}$ involves a division by τ. In a computer dividing by zero generates an infinite, according to the rules established by the IEEE Standard for Floating-Point Arithmetic (also known as IEEE 754). Also according to IEEE 754 standard, infinities have sign and the exponential of a negative infinite leads to a zero. As a consequence for $\tau = 0$ the subroutine gh_init will set p to zero which the correct result. Furthermore for $p = 0$, the coefficients beta and gamma are zero so that f2 is zero and the random number v is of no use. Then gh is equal to f1 which is $\sqrt{h}u_i$, namely a numerical realization of white noise. Therefore the function gh works fine for $\tau = 0$.

A different question is the consideration of a small but finite τ. Double precision numbers use 53 bits for the mantissa, which corresponds to about 16 significant

digits in decimal notation. The important point to be considered is that to calculate the coefficients β and γ one has to evaluate $1-p$. For $p < 2^{-53}$ the numerical evaluation of `1.d0-p` will give `1.d0` since there are not enough bits in the mantissa to evaluate the difference. This is a complete loss of accuracy and it occurs for $h/\tau < 53 \log 2 \approx 36.7$. If, for instance, one wants to evaluate $1-p$ with at least 8 digits of accuracy, then p can not be smaller than 10^{-8}, which implies $h/\tau < 8 \log 10 \approx 18.4$. Therefore, while `gh` will provide correct results for $\tau = 0$, for a finite, small τ in order to preserve accuracy it is not advisable to use time steps h larger than 18 times the correlation time τ.

7.4 Runge–Kutta-Type Methods

We consider again a stochastic differential equation with Gaussian white noise (7.1) and we would like to develop a method inspired by the Runge–Kutta methods used for numerical integration of ordinary differential equations. There is a large family of Runge–Kutta methods. Here, we focus on one of the simplest methods of this family, known as the *Heun method*, which is of second order. We start revisiting the deduction of the Heun numerical algorithm for ordinary differential equations and then we extend the ideas to the case of stochastic differential equations. So, let us consider the ordinary differential equation

$$\dot{x}(t) = q(t, x) \tag{7.92}$$

where, in general, $q(t, x(t))$ is a nonlinear function of $x(t)$ which can also depend on the time explicitly. We start by applying the Euler method to integrate this equation:

$$x(t_{i+1}) = x(t_i) + hq(t_i, x(t_i)) + O[h^2]\,. \tag{7.93}$$

This is an explicit map since the right-hand side depends only on known values of the variable x at previous times, while the unknown variable $x(t_{i+1})$ is explicitly isolated in the left-hand side. In fact, (7.93) corresponds to the explicit Euler, also called the *forward Euler method*. There is also an implicit Euler method (also called the *backward Euler method*) to integrate the same equation, which is given by

$$x(t_{i+1}) = x(t_i) + hq(t_{i+1}, x(t_{i+1})) + O[h^2]\,. \tag{7.94}$$

The implicit Euler method is more stable than the explicit one but it has the disadvantage that, in general, it is difficult to invert the function $q(t, x(t_{i+1}))$ appearing on the right-hand side so that one could get an explicit expression for $x(t_{i+1})$ as a function of $x(t_i)$. Finally, there is a semi-implicit Euler method that combines both approaches and which is of order $O[h^3]$. It is given by

$$x(t_{i+1}) = x(t_i) + \frac{h}{2}\left[q(t_i, x(t_i)) + q(t_{i+1}, x(t_{i+1}))\right] + O[h^3]\,. \tag{7.95}$$

This method suffers from the same inconvenience as the implicit Euler: the recurrence relation is given by an implicit map. A way to circumvent this inconvenience

is to replace $x(t_{i+1})$ on the right-hand side by the value obtained by using the explicit Euler method (7.93), namely by $x(t_i) + hq(t, x(t_i))$:

$$x(t_{i+1}) = x(t_i) + \frac{h}{2}\left[q(t_i, x(t_i)) + q(t_{i+1}, x(t_i) + hq(t, x(t_i))\right] + O[h^3]\,. \tag{7.96}$$

This is the Heun method that is usually written as

$$\begin{aligned} k &= hq(t_i, x(t_i)),\\ x(t_{i+1}) &= x(t_i) + \frac{h}{2}\left[q(t_i, x(t_i)) + q(t_{i+1}, x(t_i) + k)\right]\,. \end{aligned} \tag{7.97}$$

We now apply the same ideas to a stochastic differential equation with white noise (7.1). Let us start with a semi-implicit version of Euler's method given by (7.38):

$$\begin{aligned} x(t_{i+1}) = x(t_i) &+ \frac{h}{2}\left[q(t_i, x(t_i)) + q(t_{i+1}, x(t_{i+1}))\right]\\ &+ \frac{h^{1/2}u_i}{2}\left[g(t_i, x(t_i)) + g(t_{i+1}, x(t_{i+1}))\right]\,. \end{aligned} \tag{7.98}$$

And now replace $x(t_{i+1})$ on the right-hand side by the prediction given by the Euler-Maruyama algorithm (7.38). The resulting algorithm can be written as

$$\begin{aligned} k &= hq(t_i, x(t_i)),\\ l &= \sqrt{h}u_i g(t_i, x(t_i)),\\ x(t_{i+1}) &= x(t_i) + \frac{h}{2}\left[q(t_i, x(t_i)) + q(t_{i+1}, x(t_i) + k + l)\right]\\ &\quad + \frac{h^{1/2}u_i}{2}\left[g(t_i, x(t_i)) + g(t_{i+1}, x(t_i) + k + l)\right]\,. \end{aligned} \tag{7.99}$$

This is known as the *stochastic Heun method*. To study the order of convergence of this method, one can Taylor-expand the functions $q(t_{i+1}, x(t_i) + k + l)$ and $g(t_i, x(t_i)) + g(t_{i+1}, x(t_i) + k + l)$:

$$\begin{aligned} q(t_i + h, x(t_i) + k + l) &= q(t_i, x(t_i)) + l\left.\frac{\partial q(t,x)}{\partial x}\right|_{t_i, x(t_i)} + O[h],\\ g(t_i + h, x(t_i) + k + l) &= g(t_i, x(t_i)) + l\left.\frac{\partial g(t,x)}{\partial x}\right|_{t_i, x(t_i)} + O[h]. \end{aligned} \tag{7.100}$$

Substituting this into (7.99), one gets

$$\begin{aligned} x(t_{i+1}) &= x(t_i) + hq(t_i, x(t_i)) + h^{1/2}u_i g(t_i, x(t_i)) + l\frac{h^{1/2}u_i}{2}\left.\frac{\partial g}{\partial x}\right|_{t_i, x(t_i)} + O[h^{3/2}]\\ &= x(t_i) + hq(t_i, x(t_i)) + g(t_i, x(t_i))h^{1/2}u_i\\ &\quad + \frac{1}{2}g(t_i, x(t_i))g'(t_i, x(t_i))hu_i^2 + O[h^{3/2}]. \end{aligned} \tag{7.101}$$

This is precisely the Milshtein algorithm given by (7.30). The differences between the two methods are in the terms of order $h^{3/2}$ or higher. From a stochastic point of view, the Heun algorithm is of no advantage compared to the Milshtein method. The real advantage of the Heun method is that it treats better the deterministic part (the convergence of the deterministic part is of order h^3).

We can apply a similar procedure to stochastic differential equations with Ornstein–Uhlenbeck noise (7.71). In this case, the Heun algorithm is given by

$$\begin{aligned} k &= hq(t_i, x(t_i)), \\ l &= g_h(t_i)g(t_i, x(t_i)), \\ x(t_{i+1}) &= x(t_i) + \frac{h}{2}\left[q(t_i, x(t_i)) + q(t_{i+1}, x(t_i) + k + l)\right] \\ &\quad + \frac{g_h(t_i)}{2}\left[g(t_i, x(t_i)) + g(t_{i+1}, x(t_i) + k + l)\right] \end{aligned} \tag{7.102}$$

which has the same form as the algorithm for white noise replacing $\sqrt{h}u_i$ by $g_h(t_i)$. By performing a Taylor expansion of $q(t_{i+1}, x(t_i) + k + l)$ and $g(t_{i+1}, x(t_i) + k + l)$, one recovers the algorithm (7.73). The process $g_h(t)$ has to be generated as explained in Section 7.3, namely either approximating it as an Ornstein–Uhlenbeck process as given by (7.69) and then generating the Ornstein–Uhlenbeck process following (7.58), or by exact generation as given by (7.88) and (7.91). As indicated earlier, in numerical algorithms for the integration of stochastic differential equations with Ornstein–Uhlenbeck processes, typically the integration time step h must be smaller than the correlation time τ, so one cannot take naively the limit of correlation time going to zero. However, in the Heun method, if one generates $g_h(t)$ exactly, this is possible since (7.99) and (7.102) have the same form except for the noise term, and, as it was shown in Section 7.3, for $\tau \to 0$, $g_h(t_i) \to \sqrt{h}u_i$. Therefore, the Heun method for stochastic differential equations with Ornstein–Uhlenbeck noise (7.102) goes smoothly to the Heun method for stochastic differential equations with white noise (7.99). We remark that this result is valid only if the process $g_h(t)$ is generated exactly, as the approximation (7.69) requires $h \ll \tau$.

We finally discuss the computer implementation of the Heun algorithm. For white noise, a step of the algorithm can be written as

```
uh=h_sqrt*ran_g()
k=h*q(t,x)
l=uh*g(t,x)
x=x+0.5d0*(h*(q(t,x)+q(t+h,x+l+k))+uh*(g(t,x)+g(t+h,x+l+k)))
```

The functions `q(t,x)` and `g(t,x)` should be provided externally. Written in this form, at each time step, the functions `q` and `g` are called *twice with arguments* `t`, `x`, and once with arguments `t+h`, `x+k+l`. It is more efficient to write the same piece of code as

```
uh=h_sqrt*ran_g()
aux=h*q(t,x)+uh*g(t,x)
x=x+0.5d0*(aux+h*q(t+h,x+aux)+uh*g(t+h,x+aux))
```

which avoids calling functions `q` and `g` twice with the same argument.

This is a simple program to generate a trajectory of a stochastic differential equation with white noise using the Heun algorithm.

```
program Heun_white_noise
implicit none
double precision :: x,t,x_0,t_0,h,h_sqrt,uh,aux
double precision :: q,g,ran_g
integer :: i_step,n_step,i_write,n_write
external q,g,ran_g

x_0=1.d0
t_0=0.d0
h=0.01d0
n_write=200
n_step=10

h_sqrt=sqrt(h)

x=x_0
t=t_0
do i_write=1,n_write
  do i_step=1,n_step
    uh=h_sqrt*ran_g()
    aux=h*q(t,x)+uh*g(t,x)
    x=x+0.5d0*(aux+h*q(t+h,x+aux)+uh*g(t+h,x+aux))
    t=t+h
  enddo
  write (20,*) t,x
enddo

end program Heun_white_noise
```

For stochastic differential equations with Ornstein–Uhlenbeck noise, we can use a similar program replacing `uh` by the numerical generation of g_h. The following example implements the exact generation of g_h:

```
program Heun_OU_noise
implicit none
double precision :: x,t,x_0,t_0,h,tau,uh,aux
double precision :: q,g,gh
integer :: i_step,n_step,i_write,n_write
external q,g,gh

x_0=1.d0
t_0=0.d0
h=0.01d0
tau=0.5d0
n_write=200
n_step=10
```

```
x=x_0
t=t_0
call gh_init(tau,h)
do i_write=1,n_write
  do i_step=1,n_step
    uh=gh()
    aux=h*q(t,x)+uh*g(t,x)
    x=x+0.5d0*(aux+h*q(t+h,x+aux)+uh*g(t+h,x+aux))
    t=t+h
  enddo
  write (20,*) t,x
enddo
```

```
end program Heun_OU_noise
```

where the function gh returning a random number calculated according to (7.90) has been given in subsection 7.3.1.

This program can be used for correlation times τ smaller than the integration time step h, which is a big advantage compared to other methods.

The program Heun_OU_noise works fine for $\tau = 0$, and it will give the same results as Heun_white_noise in a statistical sense. They will not give exactly the same trajectories because at each time step Heun_OU_noise calls the function gh which requires the evaluation of two Gaussian random numbers (although only one is used) instead of one, and therefore the sequence of random numbers used by the two programs will be different. Besides, as Heun_OU_noise makes more calls to the random number generator and performs some additional calculations, it will be slightly slower.

While Heun_OU_noise will provide correct results for $\tau = 0$, for a finite, small τ as discussed in subsection 7.3.1 in order to preserve accuracy in the function g_h, it is not advisable to use time steps h larger than 18 times the correlation time τ. Of course, besides this technical limitation, the time step is also limited by the accuracy of the algorithm (7.102), which depends on the dynamics of the system.

7.5 Numerical Integration of Stochastic Differential Equations with Several Variables

Let us consider the following stochastic differential equation

$$\dot{x}_j(t) = q_j(t, \vec{x}) + \sum_{k=1}^{M} g_{jk}(t, \vec{x})\xi_k(t) \tag{7.103}$$

for a set of variables $\vec{x} = (x_1, \ldots, x_N)$, where $\xi_k(t)$ for $k = 1, \ldots, M$ are uncorrelated Gaussian white noises, namely

$$\begin{aligned} \langle \xi_k(t) \rangle &= 0, \\ \langle \xi_k(t)\xi_l(t') \rangle &= \delta_{kl}\delta(t - t'). \end{aligned} \tag{7.104}$$

We are interested in a numerical algorithm to generate trajectories. To do so, we proceed as in Section 7.1, namely we discretize the time and we write a formal solution of (7.103) integrating over one time step:

$$x_j(t_{i+1}) = x_j(t_i) + \int_{t_i}^{t_{i+1}} q_j(s, \vec{x}(s))ds + \sum_{k=1}^{M} \int_{t_i}^{t_{i+1}} g_{jk}(s, \vec{x}(s))\xi_k(s)ds\,. \tag{7.105}$$

For the integral involving the drift, one has

$$\int_{t_i}^{t_{i+1}} q_j(s, \vec{x}(s))ds = q_j(t_i, \vec{x}(t_i))h + O[h^{3/2}]\,. \tag{7.106}$$

The integrals involving the diffusion terms can be evaluated using the Stratonovich definition for the stochastic integral (6.46):

$$\int_{t_i}^{t_{i+1}} g_{jk}(s, \vec{x}(s))\xi_k(s)ds = g_{jk}\left(\frac{t_i + t_{i+1}}{2}, \frac{\vec{x}(t_i) + \vec{x}(t_{i+1})}{2}\right) w_{hk}(t_i) \tag{7.107}$$

where

$$w_{hk}(t_i) \equiv \int_{t_i}^{t_{i+1}} \xi_k(s)ds \tag{7.108}$$

is a Gaussian random number of zero mean and variance h. Expanding $g_{jk}(x(s))$ in a Taylor series around $\vec{x}(t) = \vec{x}(t_i)$ up to order $O[h^{1/2}]$, one has

$$g_{jk}\left(\frac{t_i+t_{i+1}}{2}, \frac{\vec{x}(t_i)+\vec{x}(t_{i+1})}{2}\right) = g_{jk}(t_i, \vec{x}(t_i)) + \sum_{l=1}^{N} \left.\frac{\partial g_{jk}(t, \vec{x})}{\partial x_l}\right|_{t_i, \vec{x}(t_i)} \frac{x_l(t_{i+1}) - x_l(t_i)}{2} + O[h]. \tag{7.109}$$

At the lowest order, $\vec{x}(t_{i+1}) - \vec{x}(t_i)$ is given by

$$x_l(t_{i+1}) - x_l(t_i) = \sum_{m=1}^{M} g_{lm}(t_i, \vec{x}(t_i))w_{hm}(t_i) + O\left[h\right]\,. \tag{7.110}$$

Substituting this into (7.109) and the result into (7.107), and using (7.106), one has

$$x_j(t_{i+1}) = x_j(t_i) + hq_j(t_i, \vec{x}(t_i)) + \sum_{k=1}^{M} g_{jk}(t_i, \vec{x}(t_i))w_{hk}(t_i) + \frac{1}{2}\sum_{k=1}^{M}\sum_{l=1}^{N}\sum_{m=1}^{M} \left.\frac{\partial g_{jk}(s, \vec{x})}{\partial x_l}\right|_{t_i, \vec{x}(t_i)} g_{lm}(t_i, \vec{x}(t_i))w_{hk}(t_i)w_{hm}(t_i) + O\left[h^{3/2}\right]\,, \tag{7.111}$$

which is the Milshtein algorithm for stochastic differential equations with several variables driven by Gaussian white noises. The Gaussian random variables $w_{hm}(t_i)$ can be written as

$$w_{hk}(t_i) = \sqrt{h}u_{k,i} \tag{7.112}$$

where $\{u_{k,i}\}$ is a set of independent Gaussian numbers of zero mean and variance 1, which can be generated using the methods discussed in Chapter 3.

It is also possible to implement a multidimensional Heun method. Starting from the multidimensional equivalent to the semi-implicit Euler method given by (7.98)

$$x_j(t_{i+1}) = x_j(t_i) + \frac{h}{2}\left[q_j(t_i, \vec{x}(t_i)) + q_j(t_{i+1}, \vec{x}(t_{i+1})\right] + \frac{1}{2}\sum_{k=1}^{M}\left[g_{jk}(t_i, \vec{x}(t_i)) + g_{jk}(t_{i+1}, \vec{x}(t_{i+1})\right] w_{hk}(t_i)\,, \tag{7.113}$$

and replacing $\vec{x}(t_{i+1})$ on the right-hand side by the prediction given by the explicit Euler algorithm, we have

$$\begin{aligned} k_j &= h q_j(t_i, \vec{x}(t_i)), \\ l_j &= \sum_{k=1}^{M} g_{jk}(t_i, \vec{x}(t_i)) w_{hk}(t_i), \\ x_j(t_{i+1}) &= x_j(t_i) + \frac{h}{2}\left[q_j(t_i, \vec{x}(t_i)) + q_j(t_{i+1}, \vec{x}(t_i) + \vec{k} + \vec{l})\right] \\ &\quad + \frac{1}{2}\sum_{k=1}^{M}\left[g_{jk}(t_i, \vec{x}(t_i)) + g_{jk}(t_{i+1}, \vec{x}(t_i) + \vec{k} + \vec{l})\right] w_{hk}(t_i) \end{aligned} \tag{7.114}$$

where $w_{hk}(t_i)$ can be generated using (7.112).

Similar ideas can be applied to stochastic differential equations with several variables driven by colored noises, of the form

$$\dot{x}_j(t) = q_j(t, \vec{x}) + \sum_{k=1}^{M} g_{jk}(t, \vec{x}) \xi_k^{\mathrm{ou}}(t) \tag{7.115}$$

where $\xi_k^{\mathrm{ou}}(t)$ are Ornstein–Uhlenbeck processes which are uncorrelated among them and with different autocorrelation times

$$\langle \xi_k^{\mathrm{ou}}(t) \rangle = 0, \tag{7.116}$$

$$\langle \xi_k^{\mathrm{ou}}(t) \xi_l^{\mathrm{ou}}(s) \rangle = \delta_{kl} \frac{1}{2\tau_k} \mathrm{e}^{-|t-s|/\tau_k}. \tag{7.117}$$

For this system, we can use the Heun algorithm (7.114) replacing $w_{hk}(t_i)$ by $g_{hk}(t_i)$, where $g_{hk}(t_i)$ is a set of Gaussian noises of zero mean and correlations given by

$$\langle g_{hk}(t_i) g_{hl}(t_i) \rangle = \delta_{kl} \left[h + \tau_k \left(\mathrm{e}^{-h/\tau_k} - 1\right)\right], \tag{7.118}$$

$$\langle g_{hk}(t_i) g_{hl}(t_j) \rangle = \delta_{kl} \tau_k \left[\frac{\mathrm{e}^{h/\tau_k} + \mathrm{e}^{-h/\tau_k}}{2} - 1\right] \mathrm{e}^{|t_i - t_j|/\tau_k}. \tag{7.119}$$

The different $g_{hk}(t_i)$ are uncorrelated, and each of them can be generated generalizing the recursion relation (7.90):

$$\begin{aligned} f_{1k}(t_{i+1}) &= \alpha u_{k,i+1}, \\ f_{2k}(t_{i+1}) &= \beta_k u_{k,i+1} + \gamma_k v_{k,i+1}, \\ g_{hk}(t_{i+1}) &= p g_{hk}(t_i) - p f_{1k}(t_i) + f_{1k}(t_{i+1}) - f_{2k}(t_i) + f_{2k}(t_{i+1}) \end{aligned} \tag{7.120}$$

where $u_{k,i}$ and $v_{k,i}$ are sets of independent Gaussian numbers with zero mean and variance 1, whereas the parameters α_k, β_k, and γ_k are given by

$$\alpha = \sqrt{h}, \tag{7.121}$$

$$\beta_k = -\frac{\tau_k(1 - p_k)}{\sqrt{h}}, \tag{7.122}$$

$$\gamma_k = \sqrt{\frac{\tau_k(1 - p_k)}{2}\left[1 - \frac{2\tau_k}{h} + p_k\left(1 + \frac{2\tau_k}{h}\right)\right]} \tag{7.123}$$

with

$$p_k \equiv \mathrm{e}^{-h/\tau_k}\,. \tag{7.124}$$

We would like to note that this method allows in a natural way the integration of stochastic differential equations driven by several noises with different correlation times. If part of the M noises are Ornstein–Uhlenbeck, say $\xi_1^{\mathrm{ou}}, \ldots, \xi_Q^{\mathrm{ou}}$, and the others are white, say $\xi_{Q+1}, \ldots, \xi_M$, then one can use the Heun algorithm with $g_{h1}, \ldots, g_{hQ}, w_{hQ+1}, \ldots, w_{hM}$ as noise terms. Besides, as discussed in Section 7.4, in the limit $\tau \to 0$, the Heun method with the exact generation for the g_h process gives the correct result for white noise, thus one can also use this method to explore the limit in which some of the noises become white.

The following example implements the Heun method for Ornstein–Uhlenbeck noises with exact generation of g_h.

```
module pars_ghv
  implicit none
  integer, parameter :: n_noises=2
  double precision, dimension (n_noises), save :: p,beta,gamma, &
                                    f1_old,f2_old,gh_aux
  double precision, save :: h_sqrt
end module pars_ghv


program Heun_OU_noise_multivariable
use pars_ghv
implicit none
interface
  function ghv()
    use pars_gvh
    double precision, dimension (n_noises) :: ghv
  end function ghv
end interface
integer, parameter :: n_var=3
double precision, dimension (nvar) :: x,x_0,qvec,qvec2,aux
double precision, dimension (n_noises) :: tau,uh
double precision, dimension (nvar,n_noises) :: gmat,gmat2
double precision :: t,t_0,h,gh
integer :: i_step,n_step,i_write,n_write,n_var,n_noises,i,j

x_0=1.d0
t_0=0.d0
h=0.01d0
do i=1,n_noises
  tau(i)=0.5d0
enddo
n_write=200
n_step=10
```

```
x=x_0
t=t_0
call ghv_init(tau,h)
do i_write=1,n_write
  do i_step=1,n_step
    uh=ghv()
    call drift (t,x,qvec,n_var)
    call diffusion (t,x,gmat,n_var,n_noises)
    aux=h*qvec + matmul(gmat,uh)
    call drift (t+h,x+aux,qvec2,n_var)
    call diffusion (t+h,x+aux,gmat2,n_var,n_noises)
    x=x+0.5d0*(aux+h*qvec2+matmul(gmat2,uh))
    t=t+h
  enddo
  write (20,*) t,x
enddo
```

```
end program Heun_OU_noise_multivariable
```

Here, `n_var` is the number of variables N and `n_noises` the number of noises M. The subroutine `drift`, to be provided by the user, should return the drift terms $\{q_i(t,\vec{x})\}$ as a vector `qq` of length N, and the subroutine `diffusion` should return the diffusion terms $\{\{g_{jk}(t,\vec{x})\}\}$ as a matrix `gg` with N rows and M columns. The function `ghv`, given below, returns a vector of independent random numbers calculated according to (7.120). The subroutine `ghv_init`, also given below, calculates the parameters and initializes the process using the computational trick (7.91). We have made use of the Fortran90 instruction `matmul` to multiply the matrix `gg` with the vector `gh`.

```
subroutine ghv_init(tau,h)
  use pars_ghv
 implicit none
 integer :: i
 double precision, dimension (n_noises) :: tau
 double precision :: h,ran_g
 h_sqrt=sqrt(h)
 p=exp(-h/tau)
 beta=-tau*(1.d0-p)/h_sqrt
 gamma= sqrt(tau*(1.d0-p)*(1.d0-2.d0*tau/h+p*(1+2.d0*tau/h))/2.d0)
 f1_old=0.d0
 do i=1,n_noises
   f2_old(i)=-sqrt(tau(i)/2.d0)*(1.d0-p(i))*ran_g()
 enddo
 gh_aux=0.d0
end function ghv_init


function ghv()
 use pars_ghv
 implicit none
 integer :: i
```

```
double precision :: ran_g
double precision, dimension (n_noises) :: ghv,u,v,f1,f2
```

```
do i=1,n_noises
  u(i)=ran_g()
  v(i)=ran_g()
enddo
f1=h_sqrt*u
f2=beta*u+gamma*v
gh_aux=p*(gh_aux-f1_old)+f1-f2_old+f2
f1_old=f1
f2_old=f2
ghv=gh_aux
```

```
end function ghv
```

7.6 Rare Events: The Linear Equation with Linear Multiplicative Noise

We can obtain very valuable information on the solution of a stochastic differential equation by looking at individual trajectories. In this section, we illustrate this by considering an apparently simple system with multiplicative noise.

$$\frac{dx}{dt} = ax(t) + \sqrt{D}x(t)\xi(t) \tag{7.125}$$

where D is a constant associated with the strength of the diffusion and $\xi(t)$ is a Gaussian white noise process of mean and correlations given by (6.39) and (6.40), respectively.

Before analyzing the stochastic differential equation (7.125), let us consider first the deterministic dynamical system

$$\frac{dx}{dt} = ax(t)\,. \tag{7.126}$$

The fixed points of the system x_{st} are determined by setting $dx/dt = 0$ in (7.126). There is only one fixed point located at $x_{st} = 0$. Now let us analyze the stability of this fixed point. To do that, we consider the time evolution of a perturbation. If the perturbation decays, the fixed point is stable; if it grows, it is unstable. Considering as the initial condition $x(0)$, the time evolution of the system is given by

$$x(t) = x(0)\mathrm{e}^{at}\,. \tag{7.127}$$

Therefore, considering $x(0)$ to be an initial condition close to the fixed point $x_{st} = 0$, that is, a small perturbation for $a < 0$, this perturbation decays, namely the fixed point is stable. In fact, for in this simple linear dynamical system, for $a < 0$ all initial conditions $x(0)$ decay to the stationary fixed point no matter how large was the initial perturbation. For $a > 0$, any small perturbation of the fixed point $x_{st} = 0$ will grow exponentially; thus, the fixed point is unstable. Therefore, there is a critical value for a at which the stability of the fixed point changes. In this system,

the critical value for a is $a_c = 0$. We note the fact that perturbations can grow to infinity, which is is an artifact of the simplicity of the model. In a real system, there will be always be saturating terms that will stop the growth of $x(t)$. For instance, a realistic equation could be one with a saturating cubic term

$$\frac{dx}{dt} = ax(t) - x(t)^3. \tag{7.128}$$

Setting $dx/dt = 0$, one finds that, for this system, the fixed points are given by

$$0 = (a - x_{\text{st}}^2)x_{\text{st}}\,. \tag{7.129}$$

As in the case of the linear system (7.126), there is a fixed point located at $x_{\text{st}} = 0$. For $a < 0$, this is the only fixed point, which is stable. For $a > 0$, there are two additional fixed points located at $x_{\text{st}} = \pm\sqrt{a}$, which can be proved to be stable. Thus, for $a > 0$, small perturbations of the unstable zero state will grow exponentially only initially. At some point, the nonlinear term will be relevant in decreasing the growth rate. Finally, the system will end up in one of the two steady states $x_{\text{st}} = \pm\sqrt{a}$.

Now we go back to the original stochastic differential equation (7.125). As it is a linear equation, it can be solved analytically as

$$\frac{dx}{x(t)} = \left[a + \sqrt{D}\xi(t)\right] dt\,. \tag{7.130}$$

Integrating both sides from $t = 0$ to time t, one has

$$x(t) = x(0)\mathrm{e}^{at+\sqrt{D}W(t)} \tag{7.131}$$

where $W(t)$ is the Wiener process. From this equation and using the properties of the Wiener process, we can compute, for instance, the evolution of the mean value of $x(t)$:

$$\langle x(t)\rangle = \langle x(0)\mathrm{e}^{at+\sqrt{D}W(t)}\rangle = x(0)\mathrm{e}^{(a+D/2)t} \tag{7.132}$$

where we have used the result $\langle \mathrm{e}^z\rangle = \mathrm{e}^{\frac{1}{2}\sigma^2[z]}$, which is valid for a Gaussian variable z of zero mean and whose proof was proposed in Exercise 1.11. From (7.131), it follows that the mean value of $x(t)$ grows exponentially to infinity for $a > -D/2$ and decays to zero for $a < -D/2$. Figure 7.1 shows the exponential growth of the mean value for $D > -2a$. Comparing this result with the one obtained in the deterministic system (7.126), one would say that the noise has shifted the threshold value for the instability of the fixed point $x_{\text{st}} = 0$ from $a_c = 0$ to $a_c = -D/2$.

However, let us look at this in more detail. The first sign that something is wrong in the previous analysis is that we could repeat the calculation for the evolution of the nth moment $\langle x(t)^n\rangle$ with the result

$$\langle x(t)^n\rangle = \left\langle \left(x(0)\mathrm{e}^{at+\sqrt{D}W(t)}\right)^n \right\rangle = x(0)^n \mathrm{e}^{n(a+nD/2)t}. \tag{7.133}$$

Therefore, the threshold for the instability of the fixed point $x_{\text{st}} = 0$ seems to be shifting to different values depending on the order of the moment used for the calculation $a_c = -nD/2$. Thus, one may ask which is the value of "real" threshold for instability of the zero solution.

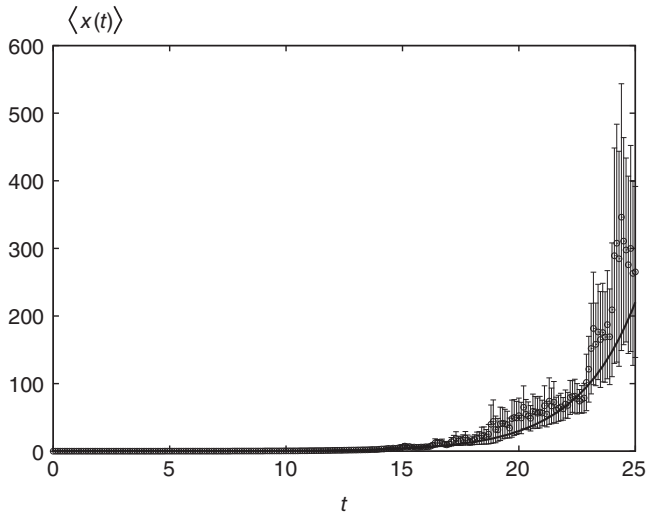


Figure 7.1 Time evolution of the mean value for the linear stochastic differential equation (7.125) for $a = -0.1$, $D = 1$, and taking as initial condition $x_0 = 0.01$. The solid line is the exact result (7.132), whereas the symbols and error bars come from numerical integration averaging over 10^6 trajectories.

If we look at the solution (7.131), we can say that the deterministic contribution at will always dominate for large time t over the stochastic contribution $\sqrt{D}W(t)$, which is of order $t^{1/2}$. Hence, for large t, and for $a < 0$, every trajectory will go to zero, and consequently $\langle x(t)^n \rangle = 0$, in contradiction with previous results. In fact, the statement that at dominates over $W(t)$ for large times is not very precise because at is a number while $W(t)$ is a stochastic process. A precise statement is that the probability that $x(t)$ decays to zero, that is, that it takes a value smaller than any number ϵ, tends to 1 as the time increases:

$$a < 0, \forall \epsilon, \quad \lim_{t \to \infty} \text{Probability}(x(t) < \epsilon) = 1. \tag{7.134}$$

As $W(t)$ has a Gaussian distribution, in principle, it has a finite probability to overcome the deterministic contribution at at any time. What the relation (7.134) indicates is that, as time increases, this probability goes to zero, and we can say, properly speaking, that every trajectory will tend to zero with probability 1. However, for any finite time, there is always a finite (although small) probability that $W(t)$ takes a large value overcoming at. This is illustrated in Figure 7.2 displaying several representative trajectories. Note the vertical scale: although every trajectory decays to zero as time increases, very large fluctuations can happen for any trajectory. Summarizing for the linear equation with multiplicative noise (7.125), there are rare trajectories (with decreasing probability as time increases) that become arbitrarily large. It is the contribution of those trajectories that makes the moments to diverge. The larger the moment, the larger is the contribution of

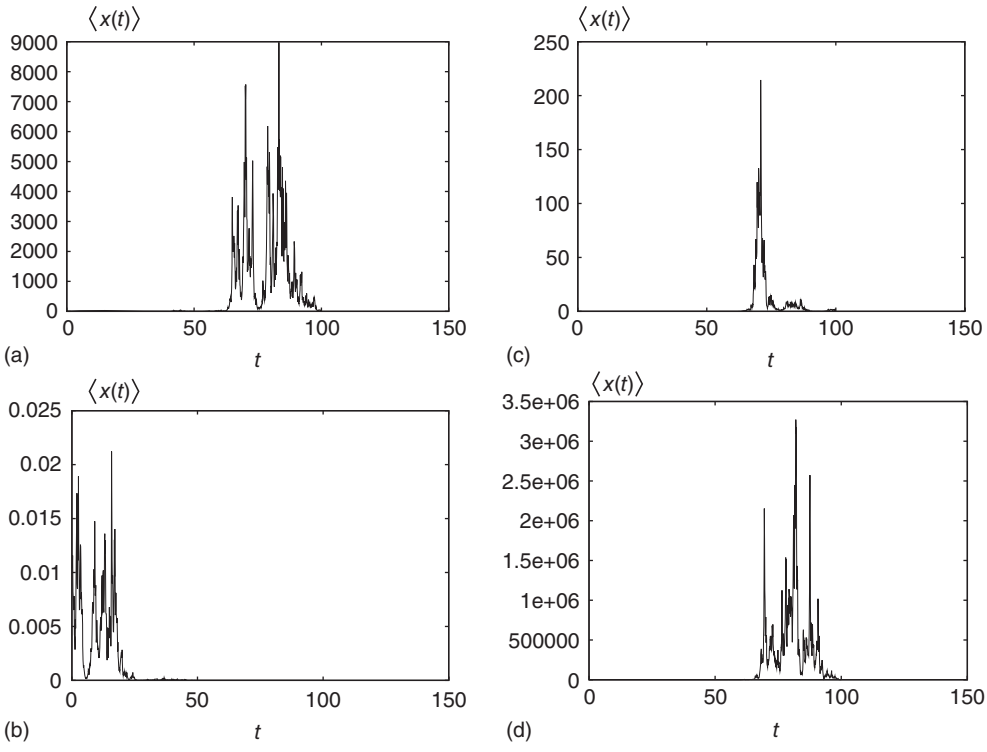


Figure 7.2 (a–d) Representative trajectories of the linear stochastic differential equation with multiplicative noise (7.125). Parameter values as in Figure 7.1.

these rare events, which explains why a_c is different depending on the moment considered.

Now, in a real system, there will always be saturating terms that will avoid the divergences of $x(t)$. For instance, we consider a cubic term as before, so that (7.125)) becomes

$$\frac{dx}{dt} = ax(t) + \sqrt{D}x(t)\xi(t) - x(t)^3 \,. \tag{7.135}$$

The nonlinear term will suppress the large fluctuations in the trajectories one by one, not just on average. Figure 7.3 shows some representative trajectories for the nonlinear stochastic differential equation (7.135) to show that, effectively, there are no large fluctuations. We conclude, then, that the mean value of any moment $\langle x(t)^n \rangle$ will tend to zero for $a < 0$. In fact, it is possible to use the Fokker–Planck equation associated to (7.135) to calculate the moments $\langle x(t)^n \rangle$ with the conclusion that all moments tend to zero for $a < 0$. Figure 7.4 displays the time evolution of the mean value obtained from a numerical integration of (7.135), showing that it tends to zero as time increases. We can say that the presence of the multiplicative noise makes no shift in the critical value for the stability of the zero solution.

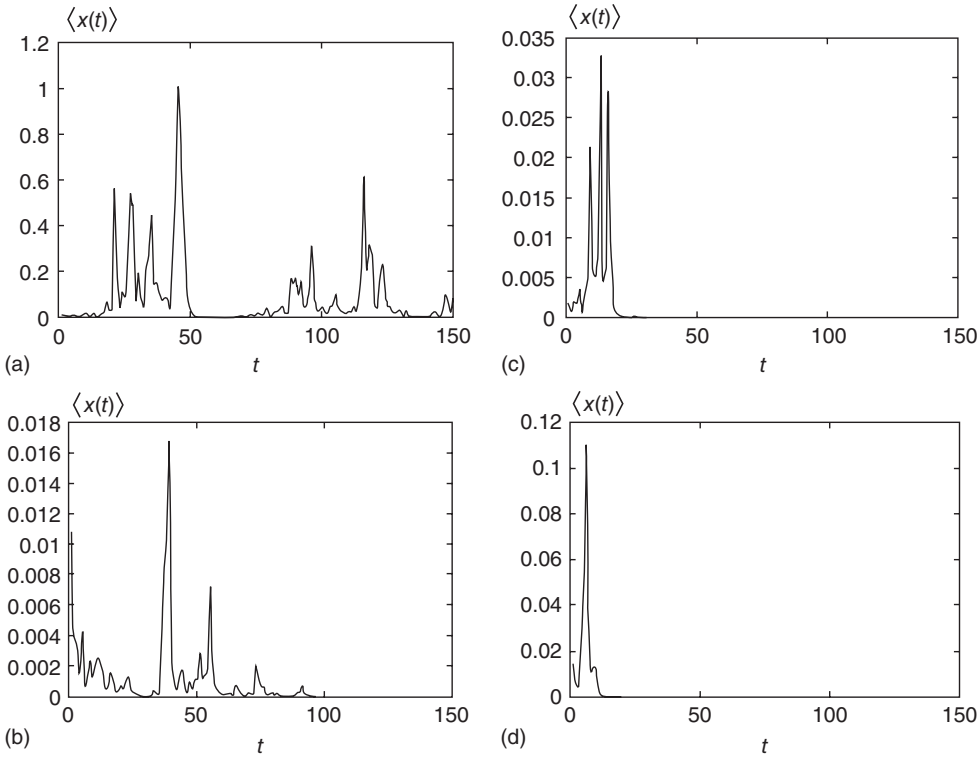


Figure 7.3 (a–d) Representative trajectories of the nonlinear stochastic differential equation with multiplicative noise (7.135). Parameter values as in Figure 7.1.

7.7
First Passage Time Problems

Sometimes, one is interested in evaluating the time it takes for a system to leave a predetermined region $\mathcal{A}$ in phase space. In general, this time will depend on the initial location of the system x_0. Even if the initial location of the system is perfectly known, for a system subject to noise fluctuations the exit time $T(x_0)$ is going to be random. Furthermore, once the system has exited $\mathcal{A}$, it may reenter it at a later time, which for systems subject to noise is also random. Thus, after a long time of evolution, the system may have exited and reentered the region $\mathcal{A}$ several times. Typically, one is interested in characterizing the statistical properties of the distribution of times T at which the system exits region $\mathcal{A}$ for the first time. This distribution of times is known as the *first passage time distribution* $P(T)$. In this section, we will illustrate how one can make use of the numerical methods introduced in the previous sections to evaluate the statistics of the first passage time. Particularly relevant are the lower order moments of the distribution, namely the mean first passage time and its variance.

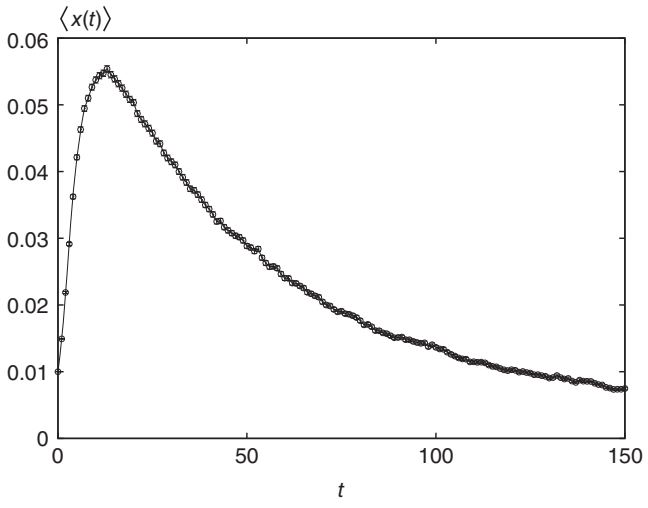


Figure 7.4 Time evolution of the mean value for the cubic stochastic differential equation (7.135) for $a = -0.1$, $D = 1$, and taking as initial condition $x_0 = 0.01$. The symbols and error bars come from numerical integration by averaging over 10^5 trajectories, and the solid line is a guide to the eye.

To fix ideas, let us consider the cubic dynamical system given by (7.128), which for convenience we write as

$$\frac{dx}{dt} = q(x) = -\frac{dV(x)}{dx} \tag{7.136}$$

where

$$V(x) = -\frac{a}{2}x^2 + \frac{1}{4}x^4 \tag{7.137}$$

is a double-well potential which we already encountered in Section 5.6. The dynamics of the system can be viewed as the movement of a particle in a potential $V(x)$ in the overdamped limit (in which the particle has no inertia). In particular, we are interested now in the case in which $a > 0$, for which the potential $V(x)$ has the shape illustrated in Figure 7.5. As discussed in Section 7.7, for $a > 0$, the system has three fixed points: one located at $x_{\mathrm{st}} = 0$, and the other two at $x_{\mathrm{st}} = \pm\sqrt{a}$. As the fixed points fulfill $q(x_{\mathrm{st}}) = 0$, they correspond to local extrema of the potential $V(x)$. The unstable fixed point $x_{\mathrm{st}} = 0$ corresponds to a maximum of the potential, whereas the two stable fixed points $x_{\mathrm{st}} = \pm\sqrt{a}$ correspond to the minima of the potential.

Now, consider that the initial state of the system at $t = 0$ is $x = 0$. Despite being unstable, because this is a fixed point, the system will remain at $x = 0$ forever. In this situation, fluctuations play a critical role because they will force the system to move out of the unstable fixed point and decay to one of the two stable fixed points. As the decay is an event triggered by noise, the time is stochastic and we are interested in characterizing its statistical properties. To be more precise, let us

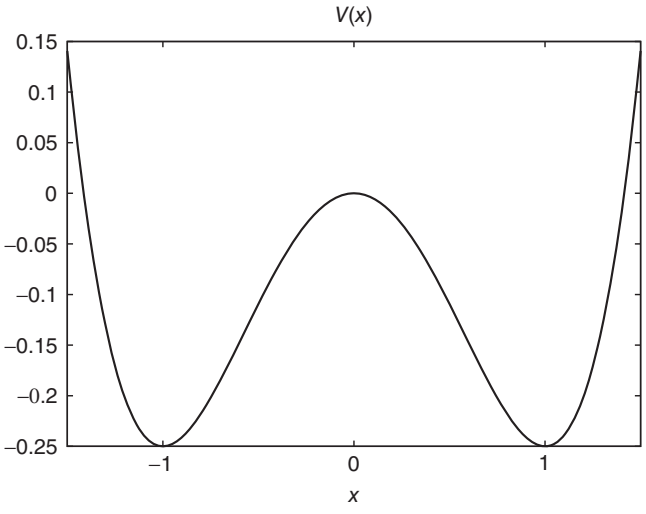


Figure 7.5 Double-well potential as given by (7.137) for $a = 1$.

assume that the system is subject to an additive noise, in which case the dynamics is described by

$$\frac{dx}{dt} = ax(t) - x(t)^3 + \sqrt{D}\xi(t) \tag{7.138}$$

where $\xi(t)$ is a Gaussian white noise with mean and correlations given by (6.39) and (6.40), respectively.[1)] It is instructive to plot several trajectories starting from the unstable fixed point. This is left as Exercise 6. We just give here a brief summary of what is observed. At $x = 0$, the deterministic contribution to the dynamics is null and the system is purely driven by noise. At some point, the trajectory $x(t)$ departs sufficiently from zero so that the linear part of the deterministic dynamics becomes relevant (while, as $|x(t)|$ is still small, the cubic term can be considered at this stage as a minor correction). The deterministic linear term pushes the system out of the origin and, as $|x(t)|$ increases, the trajectory becomes mainly dominated by the deterministic dynamics. At this stage, the noise has practically no influence in the trajectory, which basically goes monotonously to the stable fixed point. Finally, for large enough $|x(t)|$, the cubic term balances the linear term and the system settles in one of the two stable steady states. Once the stable fixed point is reached, noise makes the system to fluctuate around the fixed point.

A way to characterize the time needed for the system to decay to the steady state (known as the *decay time* from an unstable fixed point) is to consider the time it takes to reach $x = x_b$ or $x = -x_b$, where x_b is located in the region where the system decays to the fixed point in a deterministic way. This can be viewed as a first passage time problem, as follows: We can define a region $\mathcal{A}$ in phase space given

1) The main difference with respect to Eq. (7.135) in the previous section is that now the noise term does not vanish anywhere while in (7.135) at $x = 0$ there is an absorbing barrier that can not be trespassed.

by the interval $[-x_b, x_b]$. Then, starting from $x(0) = 0$, we look at the statistics of the time the system takes to cross the boundaries of the region $\mathcal{A}$ for the first time.

Numerically, the first passage time statistics can be obtained by generating many trajectories, all starting from the same initial condition $x(0) = 0$ but with different realizations of the noise. For each trajectory, we record the time it takes to reach either $x = x_b$ or $x = -x_b$. To generate the trajectories, we can use the Milshtein or the Heun algorithm discussed previously. As we are only interested in the first passage time, once the boundary is reached there is no point in continuing the numerical integration of the trajectory, so we can stop the integration and start with the next trajectory. That is, instead of integrating all the trajectories up to a given time, we integrate all the trajectories up to a given $|x_b|$. The following is an example of a program using the Heun method.

```
program Heun_first_passage_time
implicit none
double precision :: x_0=0.d0,t_0=0.d0,x_b=1.d0,h=0.01d0, &
  t_max=100.d0
double precision :: x,t,h_sqrt,uh,aux,ran_g
integer :: i_tra,n_tra=1000
double precision :: q,g,a=4.d0,D_sqrt=0.1d0
q(t,x)=a*x-x**3
g(t,x)=D_sqrt
h_sqrt=sqrt(h)

do i_tra=1,n_tra
  x=x_0
  t=t_0
  do while (abs(x) < x_b)
    uh=h_sqrt*ran_g()
    aux=h*q(t,x)+uh*g(t,x)
    x=x+0.5d0*(aux+h*q(t+h,x+aux)+uh*g(t+h,x+aux))
    t=t+h
    if (t > t_max) then
      print *,'WARNING. Maximum integration time reached.'
      print *, 'Trajectory ', i_tra, &
               ' has not crossed the boundary.'
      exit
    endif
  end do
  write (20,*) i_tra, t
enddo

end program Heun_first_passage_time
```

The core of the program is a double loop in which the Heun algorithm is applied at each time step. The outer loop is over the trajectories. The inner loop integrates a trajectory starting from `x_0` at time `t_0`, which for this problem are both zero. The `do while` loop iterates the Heun algorithm as many times as needed until $|x(t)|$ reaches `x_b`. Once the barrier is reached, the `do while` loop finishes and the program writes the time needed for the trajectory to reach the boundary to a

file. The precision in the determination on the time is given by the time step `h`. If everything goes fine, the boundary should be reached in a reasonable time. As in this case the number of iterations of a loop depends on the dynamics instead of being predetermined, it is highly advisable to set a maximum for the iterations so that if something goes wrong the program finishes after a maximum time. For instance, imagine you set the wrong sign for a, so that the system never leaves the vicinity of $x = 0$. The safeguard is here provided by a conditional that exits the inner loop if the trajectory is longer than a maximum time.

The results of the output file can be used to evaluate the mean first passage time or higher moments of the first passage time distribution. It is also possible to use them to generate a histogram of the passage time distribution (see Exercise 7).

We note that the precise value selected for the location of the barrier is not critical provided it is located in the region of deterministic dynamics. If, for instance, one chooses $x_b' > x_b$, then for each trajectory the time to reach x_b' will be slightly larger, but since the evolution from x_b to x_b' is dominated by the deterministic dynamics, the increase in time will be the same for all the trajectories. Therefore, the distribution of first passage times will be basically the same but just slightly shifted toward larger times. This is discussed in more detail in Exercise 7.

7.8 Higher Order (?) Methods

In this section, we will consider stochastic differential equations with an additive white noise of the form

$$\dot{x}(t) = q(x) + \xi(t)\,. \tag{7.139}$$

For systems with a single variable, this is not so restrictive because a stochastic differential equation of the form (7.1) can be written as one with an additive noise by introducing a new variable $y(t)$ which satisfies

$$\frac{dy}{dx} = \frac{1}{g(x)}. \tag{7.140}$$

Then, (7.1) becomes

$$\frac{dy(t)}{dt} = \frac{q(x)}{g(x)} + \xi(t). \tag{7.141}$$

That is, we have a stochastic differential equation with additive noise with an effective drift given by $q(x)/g(x)$. Unfortunately, this trick usually does not work for more than one variable.

As discussed at the beginning of this chapter, the standard numerical methods for the integration of ordinary differential equations cannot be directly used here. The main problem is that one does not know the numerical value of $\xi(t)$ at a given time, as it is a Gaussian random variable of zero mean and infinite variance. We also learn that a usual a way out of this difficulty is to integrate. In the integration process, we did not pay much attention to the form of the drift term $q(x)$. Now, let

us assume that we can separate it into two parts: a linear part of the form ωx, and a nonlinear part, which we will call $a(x)$, and let us try to improve the results by integrating exactly the linear part. So we consider a stochastic differential equation of the form

$$\dot{x} = \omega x + a(x) + \xi(t)\,. \tag{7.142}$$

The exact solution of the linear equation ($a(x) = 0$) is given by

$$x(t) = \mathrm{e}^{\omega t} x(0) + \mathrm{e}^{\omega t} \int_0^t \mathrm{e}^{-\omega s} \xi(s) ds\,. \tag{7.143}$$

This suggests the change of variables $x(t) \to z(t)$ defined as

$$x(t) = \mathrm{e}^{\omega t} z(t) + \mathrm{e}^{\omega t} \int_0^t ds \mathrm{e}^{-\omega s} \xi(s) \equiv \mathrm{e}^{\omega t} z(t) + G(t)\,. \tag{7.144}$$

Substituting (7.144) in (7.142), one finds that the variable $z(t)$ satisfies

$$\dot{z} = \mathrm{e}^{-\omega t} a(t) \tag{7.145}$$

where $a(t)$ is the value of $a(x(t))$ after replacing $x(t)$ by (7.144). Formally, it looks as if by some trick we have managed to transform a stochastic differential equation into a deterministic one. However, the noise dependence is still there because $a(t)$, as detailed before, depends on $\xi(t)$ through the relation between $x(t)$ and $z(t)$ as given by (7.144). Now comes the good news: the dependence of $z(t)$ on $x(t)$ does not involve the noise at a given time $\xi(t)$, but the process $G(t)$ which contains an integral of $\xi(t)$. And, as discussed previously, the integrals of the noise are better behaved than the noise itself. As a consequence, one may formally consider that the effect of the noise on the variable $z(t)$ is smooth in some sense, and then analyze the consequences of the application of high-order numerical methods for ordinary differential equations to (7.145). Therefore, we can formally solve (7.145) using any numerical method for ordinary differential equations.

7.8.1 Heun Method

As a first example, we consider the second-order Heun method which for ordinary differential equations is given by (7.97). For an equation of the form $\dot{z} = f(t, z)$, we rewrite it as

$$\begin{aligned} z^{(1)} &= z(t_i) + h f(t_i), \\ z(t_{i+1}) &= z(t_i) + \frac{h}{2}\left[f(t_i) + f(t_{i+1}, z^{(1)})\right]\,. \end{aligned} \tag{7.146}$$

Therefore, for (7.145), one has

$$\begin{aligned} z^{(1)} &= z(t_i) + h \mathrm{e}^{-\omega t_i} a(t_i), \\ z(t_{i+1}) &= z(t_i) + \frac{h}{2}\left[\mathrm{e}^{-\omega t_i} a(t_i) + \mathrm{e}^{-\omega t_{i+1}} a(t_{i+1})^{(1)}\right] \end{aligned} \tag{7.147}$$

where by $a(t_{i+1})^{(1)}$ we mean that the function $a(t, z)$ at time t_{i+1} is evaluated using $z^{(1)}$. Replacing $z(t) = \mathrm{e}^{-\omega t}(x(t) - G(t))$, one has

$$x^{(1)} = \mathrm{e}^{\omega h} x(t_i) + h\mathrm{e}^{\omega h} a(t_i)) + g(t_i), \tag{7.148}$$

$$x(t_{i+1}) = \mathrm{e}^{\omega h} x(t_i) + \frac{h}{2}\left[\mathrm{e}^{\omega h} a(t_i) + a(t_{i+1})^{(1)}\right] + g(t_i) \tag{7.149}$$

where we have introduced

$$g(t_i) = G(t_{i+1}) - \mathrm{e}^{\omega h} G(t_i) = \mathrm{e}^{\omega t_{i+1}} \int_{t_i}^{t_{i+1}} \mathrm{e}^{-\omega s} \xi(s) ds\,. \tag{7.150}$$

$g(t_i)$ is a Gaussian stochastic process with zero mean and correlation given by

$$\begin{aligned}\left\langle g(t_i) g(t_j) \right\rangle &= \mathrm{e}^{\omega t_{i+1}} \mathrm{e}^{\omega t_{j+1}} \int_{t_i}^{t_{i+1}} \int_{t_j}^{t_{j+1}} \mathrm{e}^{-\omega s} \mathrm{e}^{-\omega s'} \left\langle \xi(s)\xi(s') \right\rangle ds ds' \\ &= \mathrm{e}^{\omega t_{i+1}} \mathrm{e}^{\omega t_{j+1}} \int_{t_i}^{t_{i+1}} \int_{t_j}^{t_{j+1}} \mathrm{e}^{-\omega s} \mathrm{e}^{-\omega s'} \delta(s - s') ds ds'. \end{aligned} \tag{7.151}$$

For $i \neq j$, the two integrals do not overlap and the Dirac delta is always zero. For $i = j$, we have

$$\left\langle g(t_i)^2 \right\rangle = \mathrm{e}^{2\omega t_{i+1}} \int_{t_i}^{t_{i+1}} \mathrm{e}^{-2\omega s} ds = \frac{1}{2\omega}\left(\mathrm{e}^{2\omega h} - 1\right)\,. \tag{7.152}$$

Therefore, it can be generated as

$$g(t_i) = \sqrt{\frac{1}{2\omega}\left(\mathrm{e}^{2\omega h} - 1\right)} u_i \tag{7.153}$$

where u_i is a Gaussian random number of zero mean and variance 1.

Now, let us compare this version of the Heun algorithm, (7.148, 7.149, 7.153), with the one we derived in Section 7.4, namely (7.99). To do so, consider the limit $\omega \to 0$ in which case there is no splitting in linear and nonlinear parts. In this limit, $\mathrm{e}^{\omega h} \to 1$ and $\left\langle g(t_i)^2 \right\rangle \to h$, thus $g(t_i) \to \sqrt{h} u_i$. Therefore, one recovers the original algorithm.

The core of the method implementing the evolution on one time step can be written as follows:

```
a0=a(t,x)
g=eqs*ran_g()
x1=g+w*(x+h*a0)
x=g+w*x+h_half(w*a0+a(t+h,x1))
t=t+h
```

where `x` at the beginning corresponds to $x(t_i)$ and at the end to $x(t_{i+1})$, which will be used in the following time step. `x1` stores $x(t_{i+1})^{(1)}$. `w` stores $\mathrm{e}^{\omega h}$. We use `a0` to store the nonlinear term $a(t_i)$. `h_half` corresponds to $h/2$. The function `a(t,x)` should be provided externally, and it should return the nonlinear term. `eqs` is the strength of the stochastic term, $\sqrt{(\mathrm{e}^{2\omega h} - 1)/2\omega}$.

The Heun method is a second-order Runge–Kutta method. Another second-order Runge–Kutta method is the midpoint. Furthermore, the Heun method can also be viewed as composed of two stages: first it makes a (crude) approximation for the value of the variable at time t_{i+1}; then uses this result to evaluate the nonlinear

term. Pursuing this idea a little further leads to the predictor–corrector methods. In what follows, we discuss the midpoint Runge–Kutta and predictor–corrector methods.

7.8.2 Midpoint Runge–Kutta

The midpoint Runge–Kutta method for an equation of the form $\dot{z} = f(t, z)$ reads

$$\begin{aligned} z^{(1)} &= z_i + \frac{h}{2} f(t_i, z(t_i)), \\ z(t_{i+1}) &= z(t_i) + h f(t_i + h/2, z^{(1)}) + O[h^3]\,. \end{aligned} \tag{7.154}$$

We can now apply the method to (7.145):

$$\begin{aligned} z^{(1)} &= z(t_i) + \frac{h}{2} \mathrm{e}^{-\omega t_i} a(t_i), \\ z(t_{i+1}) &= z(t_i) + h \mathrm{e}^{-\omega(t_i + h/2)} a(t_i + h/2)^{(1)} \end{aligned} \tag{7.155}$$

where by $a(t_i + h/2)^{(1)}$ we mean that the function $a(t, z)$ at time $t_i + h/2$ is evaluated using $z^{(1)}$. Replacing back $x(t)$ using (7.144), it can be shown to reduce to

$$\begin{aligned} x^{(1)} &= \mathrm{e}^{\omega h/2} \left(x(t_i) + \frac{h}{2} a(t_i) \right) + g_1(t_i), \\ x(t_{i+1}) &= g(t_i) + \mathrm{e}^{\omega h} x(t_i) + h \mathrm{e}^{\omega h/2} a(t_i + h/2)^{(1)} \end{aligned} \tag{7.156}$$

where

$$g_1(t_i) = G\left(t_i + \frac{h}{2}\right) - \mathrm{e}^{\omega h/2} G(t_i) = \mathrm{e}^{\omega(t_i + h/2)} \int_{t_i}^{t_i + h/2} \mathrm{e}^{-\omega s} \xi(s) ds, \tag{7.157}$$

$$g(t_i) = G(t_i + h) - \mathrm{e}^{\omega h} G(t_i) = \mathrm{e}^{\omega h/2} g_1(t_i) + g_1\left(t_i + \frac{h}{2}\right). \tag{7.158}$$

Notice that $g_1(t_i)$ and $g_1(t_i + \frac{h}{2})$ are independent zero-mean Gaussian variables and that $\langle g_1(t_i) g_1(t_j) \rangle = \langle g_1(t_i)^2 \rangle \delta_{i,j}$ for $t_i = ih$ so

$$\langle g_1(t_i)^2 \rangle = \sqrt{\frac{\epsilon}{\omega} \left(\mathrm{e}^{\omega h} - 1 \right)}. \tag{7.159}$$

A disadvantage of the midpoint method with respect to the Heun method (7.149) is that now we have to generate two random numbers instead of one at each time step.

7.8.3 Predictor–Corrector

As the name indicates, the predictor–corrector method is composed of two stages. First, one predicts the value for the variable at the next integration time step using a polynomial extrapolation. The polynomial is determined by the values of the variable and its derivatives at previous integration time steps. In the corrector

stage, one uses this extrapolated value to determine the derivative at the next time step, and adjusts another polynomial using that derivative and the values of the variable or its derivatives at previous integration time steps. There are several predictor–corrector methods depending on the degree of the polynomials and which values in the past are used to determine them. As predictor–corrector methods involve the values of the variable or its derivative in several previous time steps, they are multistep methods, while methods such as Euler or Runge–Kutta, which refer to the values of the variable and its derivative only at one previous time, are classified as single-step methods.

Here we will make use of the Adams–Bashford–Moulton fourth-order method. The predictor is the Adams–Bashford extrapolation, which for a differential equation of the form $\dot{z} = f(t, z)$ reads

$$z(t_{i+1})^p = z(t_i) + \frac{h}{24}[55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}] + O[h^5] \tag{7.160}$$

where f_i stands for $f(t_i, z(t_i))$ and we have used the super-index p to indicate that $z(t_{i+1})^p$ is the value of z at time t_{i+1} given by the predictor. Note that the predictor is a four-step method; that is, to predict the next step, it uses the value of the variable and the derivative at the present time, and the value of the derivative in the three preceding time steps.

Once the value of the variable is predicted, then it is used by the corrector. The corrector is the Adams–Moulton method, given by

$$z(t_{i+1}) = z(t_i) + \frac{h}{24}[9f^p_{i+1} + 19f_i - 5f_{i-1} + f_{i-2}] + O[h^5] \tag{7.161}$$

where f^p_{i+1} stands for $f(t_{i+1}, z(t_{i+1})^p)$. If one did not have the value $z(t_{i+1})^p$ given by the predictor to calculate f_{i+1} in the right-hand side of (7.161), the evaluation of the corrector would have required solving an implicit equation for $z(t_{i+1})$.

Applying (7.160) and (7.161) to (7.145), we have

$$\begin{aligned} z(t_{i+1})^p = z(t_i) + \frac{h}{24}\Big[&55\mathrm{e}^{-\omega t_i}a(t_i) - 59\mathrm{e}^{-\omega t_{i-1}}a(t_{i-1}) \\ &+37\mathrm{e}^{-\omega t_{i-2}}a(t_{i-2}) - 9\mathrm{e}^{-\omega t_{i-3}}a(t_{i-3})\Big] \end{aligned} \tag{7.162}$$

and

$$\begin{aligned} z(t_{i+1}) = z(t_i) + \frac{h}{24}\Big[&9\mathrm{e}^{-\omega t_{i+1}}a(t_{i+1})^p + 19\mathrm{e}^{-\omega t_i}a(t_i) \\ &-5\mathrm{e}^{-\omega t_{i-1}}a(t_{i-1}) + \mathrm{e}^{-\omega t_{i-2}}a(t_{i-2})\Big]. \end{aligned} \tag{7.163}$$

By replacing $z(t) = \mathrm{e}^{-\omega t}\,(x(t) - G(t))$, we obtain

$$\begin{aligned} x(t_{i+1})^p = g(t_i) + \mathrm{e}^{h\omega}\Bigg[&x(t_i) + \frac{h}{24}\left(55a(t_i) - 59\mathrm{e}^{h\omega}a(t_{i-1})\right. \\ &\left. +\ 37\mathrm{e}^{2h\omega}a(t_{i-2}) - 9\mathrm{e}^{3h\omega}a(t_{i-3})\right)\Bigg] \end{aligned} \tag{7.164}$$

and

$$\begin{aligned} x(t_{i+1}) = g(t_i) + \mathrm{e}^{h\omega}x(t_i) + \frac{h}{24}\Big[&9a(t_{i+1})^p + 19\mathrm{e}^{h\omega}a(t_i) \\ &-5\mathrm{e}^{2h\omega}a(t_{i-1}) + \mathrm{e}^{3h\omega}a(t_{i-2})\Big]. \end{aligned} \tag{7.165}$$

The core of the method implementing the evolution on one time step can be written as follows:

```
a0=a(t,x)
g=eqs*ran_g()
v=g+w*(u+dt55*a0-dt59*a1+dt37*a2-dt9*a3)
ap=a(v)
u=g+dt9*ap+w*(u+dt19*a0-dt5*a1+dt1*a2)
a3=a2*w
a2=a1*w
a1=a0*w
t=t+h
```

where `v` stores $x(t_{i+1})^p$ and `w` stores $e^{\omega h}$. We use `a0`, `a1`, `a2`, and `a3` to store the nonlinear terms $a(t_i)$, $e^{\omega h}a(t_{i-1})$, $e^{2\omega h}a(t_{i-2})$, and $e^{3\omega h}a_k(t_{i-3})$, respectively. At the end of the step, these numbers are multiplied by `w` so that `a2` is converted to `a3`, and so on. `ap` stores $a(t_{i+1})^p$. The coefficient `dt55` corresponds to $55h/24$, and so on. As in the Heun method discussed in subsection 7.8.1, `a(t,x)` is an external function returning the nonlinear term and eqs is the strength of the stochastic term.

Regarding implementation of multistep methods, we would like to note that these methods need to be warmed up by generating the first time steps using a single-step method. The Adams–Bashford–Moulton method requires the knowledge of four time steps before it can be started. One can obtain the values of the nonlinear function at these initial steps integrating the equation using, for instance, a lower order method with a much smaller time step.

7.8.4 Higher Order?

The procedure we have just discussed allows the exact integration of the linear part of the deterministic equation and the formal use of higher order methods. Thus, these algorithms have the advantage of providing better accuracy for the deterministic part of the equation. Nevertheless, it cannot be proved that these algorithms are indeed of higher order since, for example, the predictor part of a predictor–corrector assumes a smooth function to extrapolate the value at the next time step. This is not completely fulfilled by the variable x, since at each time step a random term is added. Thus, we should consider these methods of order $O[h^{3/2}]$, which is the order of the Milshtein method, although they have the advantage giving a much more accurate description of the deterministic part.

Further Reading and References

The Milshtein method for Itô stochastic differential equations was published by Grigori N. Mil'shtein in Russian in 1974, translated to English by K. Durr [31]. For a deduction of the method for Stratonovich stochastic differential equations, see [32].

For the difficulties in extending the Milshtein method to higher order, see [33].

The exact calculation of the moments of (7.135) is performed in [34].

For an extensive review on algorithms for numerical integration of stochastic differential equations, see the book by P.E. Kloeden, and E. Platen, [35]. For a computationally oriented introduction to these algorithms, including examples of computer programs, see the book by P.E. Kloeden, E. Platen, and H. Schurz [36].

The Heun method for ordinary differential equations is named after Karl Heun. For its stochastic version, see the book by T.C. Gard [37].

The method for exact generation of the process $g_h(t)$ was proposed in [38].

Large fluctuations are particularly relevant in small systems out of thermodynamic equilibrium and has been object of intensive research in recent years, for a review see [39].

Exercises

1) Consider the stochastic differential equation $\dot{x} = -ax + \sqrt{D}\xi(t)$, where $\xi(t)$ is a Gaussian white noise defined in (6.39) and (6.40).
   a. Integrate numerically the equation for $a = 2$ and $D = 0.05$ and plot 10 trajectories starting from the initial condition $x(0) = 4$ together with the deterministic trajectory. Identify an initial dynamical regime dominated by deterministic dynamics and a stationary regime in which the system fluctuates around the steady state.
   b. Plot several trajectories starting from different initial conditions and for different values of a. Discuss the behavior; in particular, identify the time it takes for the system to reach the stationary state. How does it depend on the parameter a?
   c. Evaluate from numerical integration the moments $\langle x(t)\rangle$ and $\langle x(t)^2\rangle$, where $\langle\cdots\rangle$ stands for averages over trajectories, for $a = 2$ and $D = 0.05$ with initial condition $x(0) = 1$. Perform the averages over 10, 100, and 1000 trajectories. Plot the moments as a function of time and identify the stationary regime in which the moments do not depend on time. Discuss the influence of the number of trajectories on the averages.
   d. Evaluate from numerical integration the stationary value for the second moment $\langle x(t)^2\rangle$ for several values of D with $a = 2$ and considering as initial condition $x(0) = 0$. Plot the second moment as a function of the noise strength D and discuss the results.
   e. In the stationary regime, evaluate $\langle x(t)\rangle_t$ and $\langle x(t)^2\rangle_t$, where $\langle\cdots\rangle_t$ stands for averaging over time in a single (long) trajectory, for several values of the noise intensity D. Perform the averages over different time intervals and discuss the results. Plot the second moment as function of the noise strength and compare the results with those obtained in the previous point averaging over trajectories.
2) Consider the same stochastic differential equation as in Exercise 1.

a. Integrate the equation numerically with $a = 2$ and $D = 0.05$ and $x(0) = 1$ and evaluate the correlation function $C(t, s) = \langle x(t)x(t + s)\rangle$, where $\langle \cdots \rangle$ stands for averages over trajectories. Plot $C(t, s)$ as a function of s for $t = 1$, $t = 2$, $t = 5$, $t = 10$, and $t = 20$. Identify the stationary regime in which $C(t, s) = C(s)$, that is, the correlation function does not depend on t.

b. Generate a long trajectory for $a = 2$ and $D = 0.05$ starting from $x(0) = 0$ to decrease the transients. Evaluate the correlation function $\tilde{C}(s) = \langle x(t)x(t + s)\rangle_t$, where $\langle \cdots \rangle_t$ stands for averaging over time in a single (long) trajectory, using the methodology indicated in Appendix C. Plot $\tilde{C}(s)$ and compare it with the correlation $C(s)$ calculated in the previous point.

3) Consider the stochastic differential equation $\dot{x}(t) = -ax + \sqrt{D}\xi^{\text{ou}}(t)$ where $\xi^{\text{ou}}(t)$ is a Gaussian colored noise defined in (6.58) and (6.59).

a. Integrate numerically the equation for $a = 2$, $D = 0.05$, and $\tau = 0.1$ and plot 10 trajectories starting from $x(0) = 4.0$. Identify an initial dynamical regime dominated by deterministic dynamics and a stationary regime in which the system fluctuates around the steady state. Discuss the similarities and differences in both regimes as compared with the results obtained in Exercise 1.

b. Evaluate from numerical integration the moments $\langle x(t)\rangle$ and $\langle x(t)^2\rangle$ with initial condition $x(0) = 1$, where $\langle \cdots \rangle$ means averages over trajectories. Perform the averages over 10, 100 and 1000 trajectories. Plot the moments as a function of the time, identify the stationary regime, and discuss the influence of the number of trajectories on the averages.

c. Evaluate from numerical integration the stationary value for the second moment $\langle x(t)^2\rangle$ for several values of D with $a = 2$ and considering as initial condition $x(0) = 0$. Plot the second moment as a function of the noise strength D and discuss the results.

4) Consider the same stochastic differential equation as in Exercise 3.

a. In the stationary regime, evaluate from numerical integration, for $a = 2$, $D = 0.05$, and $\tau = 0.1$, with $x(0) = 0$, the correlation $C(t, s) = \langle x(t)x(t + s)\rangle_t$, where $\langle \cdots \rangle_t$ stands for averaging over time in a single (long) trajectory, using the methodology indicated in Appendix C. Plot $C(s)$.

b. Repeat the previous point for several values of the correlation time τ and discuss the results.

5) This exercise illustrates how in some instances it is possible to generate exact trajectories.

a. Introducing the process

$$G(t) = \mathrm{e}^{at + W(t)}$$

show that the solution of (7.135) is

$$x(t) = \frac{x(0)G(t)}{\sqrt{1 + 2x(0)^2 \int_0^t ds\, G(s)^2}}.$$

b. Use a Simpson integration scheme (and the exact generation of the Wiener process $W(t)$) to compute the integral in the denominator. This is good

enough because the process $G(t)$ is continuous and differentiable. This is an alternative method to the ones explained in the text to generate trajectories of the process $x(t)$.

c. Use this method to compute the average $\langle x(t)\rangle$, and compare your results with those of Figure 7.4.

6) This exercise illustrates the decay from an unstable steady state. Consider $\dot{x}(t) = ax - bx^3 + \sqrt{D}\xi(t)$, where $\xi(t)$ is a Gaussian white noise defined in Equations (6.39) and (6.40). Take the values of the parameters as $a = 4$, $b = 1$, $D = 0.01$, and as the initial condition $x(0) = 0$.
   a. Integrate numerically the equation and plot 20 trajectories until time $t = 6$. Identify the different dynamical regimes discussed in Section 7.7.
   b. Evaluate from numerical integration $\langle x(t)\rangle$, $\langle x(t)^2\rangle$, and $\langle x(t)^4\rangle$, where $\langle \cdots \rangle$ stands for averages over trajectories. Plot the results as a function of t.
   c. Transient anomalous fluctuations: Plot the variance of $x(t)^2$, namely $\sigma^2[x(t)^2] = \langle x(t)^4\rangle - \langle x(t)^2\rangle^2$, as a function of t. Discuss the results; in particular, give a physical interpretation of the fact that $\sigma^2[x(t)^2]$ goes through a maximum at an intermediate time.

7) This exercise illustrates the evaluation of the first passage time distribution. Consider the same equation as in Exercise 6 with the same values for the parameters and taking $x(0) = 0$ as initial condition.
   a. Integrate the equations numerically to generate 1000 trajectories, recording the time at which each trajectory reaches $|x(t)| = x_b = 0.5$, as indicated in Section 7.7.
   b. Plot a histogram of the distribution of the times. This is the first passage time distribution. Is it symmetrical? Discuss the results.
   c. Evaluate the mean of the distribution (mean first passage time) and the variance for distribution.
   d. Repeat the three previous points for $x_b = 1$. Compare the results with the ones obtained previously.

8) This exercise evaluates the dependence on the noise strength of the decay time from an unstable state. Consider the same equation as in Exercise 6 with $a = 4$, $b = 1$, and $x(0) = 0$ as the initial condition. Evaluate the mean and the variance of the first passage time taking $x_b = 0.5$ for different values of the noise strength D. Plot the results as function of D. Plot also the mean first passage time as function of $\log D$. Discuss the noise dependence of both quantities.

9) This exercise illustrates how noise can induce jumps from one stable state to another in a bistable system. Consider, again, the equation $\dot{x}(t) = ax - bx^3 + \sqrt{D}\xi(t)$, where $\xi(t)$ is a Gaussian white noise defined in Equations (6.39) and (6.40). Take the values for the parameters as $a = 0.2$, $b = 1$, and the initial condition as $x_0 = x_{st} = \sqrt{a/b}$, which is one of the stable steady states in the deterministic case.
   a. Integrate numerically the equation for a long time and plot the trajectory for $D = 0.001, 0.01, 0.1$. Discuss what is observed.

b. Construct a histogram with the values taken for the system over a long trajectory. Compare it with the stationary probability distribution obtained in Exercise 6.5.
c. Discuss how the time T to jump from one minimum to the other depends on D. Compare the results with the Kramer's law $t \approx e^{\Delta V/D}$, where ΔV is the difference of the potential between the maximum and the minimum.

10) This exercise illustrates a system with a barrier. Consider the equation $\dot{x}(t) = ax - bx^3 + \sqrt{D}x\xi(t)$, where $\xi(t)$ is a Gaussian white noise defined in Equations (6.39) and (6.40). Take the values of the parameters as $a = 0.2$, $b = 1$, and the initial condition as $x_0 = x_{st} = \sqrt{b/a}$, which is one of the stable steady states in the deterministic case.
a. Integrate numerically the equation for a long time and plot the trajectory for $D = 0.001, 0.01, 0.1$ using the Milshtein method. Discuss what is observed. Do you ever observe numerically a crossing through the barrier $x = 0$, which should not be present in the real system?
b. Perform the change of variable suggested in Section 7.8 and integrate numerically the equation using a predictor–corrector. Do the spurious crossings of the barrier still appear in this case?
c. Construct a histogram with the values taken for the system over a long trajectory. Compare it with the stationary probability distribution obtained in Exercise 6.8.

11) This exercise illustrates the stochastic resonance phenomenon. Consider the equation $\dot{x}(t) = ax - bx^3 + A\sin(\omega t) + \sqrt{D}\xi(t)$, where $\xi(t)$ is a Gaussian white noise defined in Equations (6.39) and (6.40). Take the values of the parameters as $a = 1$, $b = 1$, $\omega = 0.05$, and an arbitrary initial condition.
a. Take $D = 0$ and integrate numerically the equation for different values of A. Show that there is a threshold value A_{th} for which the system starts to jump from one steady state to another.
b. Consider $A < A_{th}$, say, $A = 0.9A_{th}$, and plot the trajectories for different noise strengths. Discuss what is observed.
c. Calculate the correlation between the forcing $A\sin(\omega t)$ and the trajectory. Plot the value of the correlation as function of the noise strength and show that there is a maximum.

8
Introduction to Master Equations

In this chapter, we briefly present the main results about master equations. They are differential equations that describe the evolution of the probabilities for Markov processes for systems that jump from one state to another in continuous time. In this sense, they are the continuous-time version of the recurrence relations for Markov chains mentioned at the end of Chapter 1. We emphasize their use in case the number of available states is discrete, as it corresponds to applications to chemical reactions, radioactive substances, epidemic spreading, and many other cases of interest. We give also a brief account of the generating function technique to solve master equations and some approximate methods of solution.

8.1
A Two-State System with Constant Rates

Let us consider now a situation in which something can switch between two states which we name ''1'' and ''2.'' There are many examples of this situation:

- A radioactive atom that has not disintegrated (state 1) and after disintegration (state 2);
- A person who is healthy (state 1) or sick (state 2);
- In a chemical reaction, say the combination of sodium and chlorine to produce salt, Na + Cl → NaCl, an unbound atom of sodium (state 1) or bound to a chlorine atom (state 2);
- A bilingual person speaking one language (state 1) or the other (state 2).

In these and many other examples, one can represent the transitions between the two states as random events that happen at some rates. We denote by $\omega(1 \to 2)$ the rate at which one particle[1)] jumps from state 1 to state 2 and assume that those transitions occur uniformly and randomly at the constant rate $\omega(1 \to 2)$. This means that there is a probability $\omega(1 \to 2)dt$ that the particle jumps from state 1 to 2 in the time interval $(t, t + dt)$.

1) For the sake of brevity, we will call the agents, systems, persons, atoms, or whatever that is making the jumps between the states, as ''particles''.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

The inverse process, that of switching from 2 to 1, might or might not occur. For example, a person can get a flu (so it goes from $1 \to 2$) but he/she usually recovers from the flu (so it goes from $2 \to 1$). A radioactive substance, however, cannot switch back from the decayed atom to the original atom. In reversible chemical reactions, the product molecule (NaCl, for instance) can be broken up into the constituent atoms. The bilingual person can be switching many times a day between the two languages. When the inverse switching does occur, its rate $\omega(2 \to 1)$ might or might not be related to the rate $\omega(1 \to 2)$. If a particle in state 1 adopts the form X and a particle in state 2 the form Y, we can indicate this process schematically by

$$X \rightleftarrows Y. \tag{8.1}$$

We will use sometimes, for brevity, the notation $\omega_{i\to j} \equiv \omega(i \to j)$ to indicate the rate at which one particle goes from state i to state j.

8.1.1 The Particle Point of View

Starting at some initial time t_0, we ask the probabilities $P_1(t)$ and $P_2(t)$ that a given particle is in state 1 or 2, respectively, at time t. Obviously, they must satisfy $P_1(t) + P_2(t) = 1$. We will derive now a differential equation for $P_1(t)$. We do that by relating the probabilities at two closed times, t and $t + dt$. In doing so, we are implicitly using the Markov assumption, as the transition rate from one state to the other during the time interval $(t, t + dt)$ does not depend at all on the previous history, but only on the state at time t. The probability $P_1(t + dt)$ that the particle is in state 1 at time $t + dt$ has two contributions: that of being in 1 at time t and not having jumped to state 2 during the interval $(t, t + dt)$, and that of being at 2 at time t and having made a jump from 2 to 1 in the interval $(t, t + dt)$. Summing up all these cases and using the rules of conditional probability, we have

$$P_1(t + dt) = P_1(t)\text{Prob(staying in 1)} + P_2(t)\text{Prob(jumping from 2 to 1)}. \tag{8.2}$$

By the definition of the rate, the probability of jumping from 2 to 1 in the time interval $(t, t + dt)$ is $\omega(2 \to 1)dt$, whereas the probability of staying in state 1 is one minus the probability of leaving state 1 in the same time interval, or $(1 - \omega(1 \to 2)dt)$. This leads to

$$P_1(t + dt) = P_1(t)[1 - \omega(1 \to 2)dt] + P_2(t)\omega(2 \to 1)dt + O(dt^2). \tag{8.3}$$

The terms of order $O(dt^2)$ could arise if the particle is in state 1 at time $t + dt$ because it was in state 1 at time t and made two jumps, one from 1 to 2 and another from 2 to 1, during the time interval $(t, t + dt)$. In this way, it could end up again in state 1, thus contributing to Prob(staying in 1). This would happen with probability $\omega(1 \to 2)dt \times \omega(2 \to 1)dt = O(dt^2)$. Similarly, there could be higher order contributions from particles jumping from 2 to 1 and then back from 1 to 2. All these multiple events happen with vanishing probability in the limit $dt \to 0$. Rearranging, and taking the limit $dt \to 0$, we get the differential equation

$$\frac{dP_1(t)}{dt} = -\omega(1 \to 2)P_1(t) + \omega(2 \to 1)P_2(t). \tag{8.4}$$

A similar reasoning leads to the equivalent equation for $P_2(t)$:

$$\frac{dP_2(t)}{dt} = -\omega(2 \to 1)P_2(t) + \omega(1 \to 2)P_1(t). \tag{8.5}$$

Equations (8.4 and 8.5) are very simple examples of *master equations*: equations for the probability that a stochastic particle that can jump between different states is in one of these states at a time t. These first-order differential equations have to be solved under the assumption of some initial conditions at some initial time t_0: $P_1(t_0)$ and $P_2(t_0)$. Note that

$$\frac{d}{dt}[P_1(t) + P_2(t)] = 0 \tag{8.6}$$

implying that $P_1(t) + P_2(t) = 1$ at all times t provided that the initial condition satisfies, as it should, $P_1(t_0) + P_2(t_0) = 1$. One defines the probability current, $J(1 \to 2)$, from state 1 to 2 as

$$J(1 \to 2) = -\omega(1 \to 2)P_1(t) + \omega(2 \to 1)P_2(t). \tag{8.7}$$

It has a negative contribution coming from those jumps leaving state 1 to go to state 2, and a positive contribution from those jumps from state 2 to state 1. A similar definition leads to $J(2 \to 1) = -J(1 \to 2)$. In terms of these probability currents, the master equations are

$$\frac{dP_1(t)}{dt} = J_1(t), \quad \frac{dP_2(t)}{dt} = J_2(t). \tag{8.8}$$

In the case of constant rates $\omega(1 \to 2)$ and $\omega(2 \to 1)$, which we are considering throughout this section, it is possible to find the explicit solution for (8.4 and 8.5):

$$\begin{aligned} P_1(t) = {} & P_1(t_0)\frac{\omega_{2\to1} + \omega_{1\to2}\mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)}}{\omega_{2\to1} + \omega_{1\to2}} \\ & + P_2(t_0)\frac{\omega_{2\to1}(1 - \mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)})}{\omega_{2\to1} + \omega_{1\to2}}, \end{aligned} \tag{8.9}$$

$$\begin{aligned} P_2(t) = {} & P_1(t_0)\frac{\omega_{1\to2}(1 - \mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)})}{\omega_{2\to1} + \omega_{1\to2}} \\ & + P_2(t_0)\frac{\omega_{1\to2} + \omega_{2\to1}\mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)})}{\omega_{2\to1} + \omega_{1\to2}}. \end{aligned} \tag{8.10}$$

From here, and using $P_1(t_0) + P_2(t_0) = 1$, we can obtain the stationary distribution as the limit $t \to \infty$:

$$P_1^{\mathrm{st}} = \frac{\omega_{2\to1}}{\omega_{2\to1} + \omega_{1\to2}}, \tag{8.11}$$

$$P_2^{\mathrm{st}} = \frac{\omega_{1\to2}}{\omega_{2\to1} + \omega_{1\to2}} \tag{8.12}$$

which are particularly simple solutions. Note that in this case the stationary distribution satisfies

$$\omega(1 \to 2)P_1^{\mathrm{st}} = \omega(2 \to 1)P_2^{\mathrm{st}}, \tag{8.13}$$

showing that in the case of two states the stationary distributions satisfy the *detailed balance condition*, equivalent to (1.145) for Markov chains.

An interesting quantity is the probability $P(i,t|j,t_0)$ that the particle is in state i at time t given that it was in state j at time t_0 for $i,j=1,2$. In general, it is difficult to compute $P(i,t|j,t_0)$ directly from the rules of the process. The reason has already been mentioned earlier: in a finite interval (t_0,t), there might have been many jumps to intermediate states and all these have to be included in the calculation of the conditional probability. For instance, to compute $P(1,t|1,t_0)$, one has to include the jumps $1\to2\to1$, in which the particle has left state 1 to go to state 2 and returned from it, *any number of times*, from zero to infinity.

Luckily, there is a simple, alternative way to proceed in the case of two states and constant rates. We can obtain $P(1,t|1,t_0)$ from $P_1(t)$ setting as an initial condition $P_1(t_0)=1,\ P_2(t_0)=0$, as we know (probability 1) that the particle is in state 1 at time t_0. Taking the explicit solution (8.9) with $P_1(t_0)=1,\ P_2(t_0)=0$, we obtain

$$P(1,t|1,t_0)=\frac{\omega_{2\to1}+\omega_{1\to2}\mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)}}{\omega_{2\to1}+\omega_{1\to2}} \tag{8.14}$$

and, of course

$$P(2,t|1,t_0)=1-P(1,t|1,t_0)=\frac{\omega_{1\to2}\left(1-\mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)}\right)}{\omega_{2\to1}+\omega_{1\to2}} \tag{8.15}$$

with equivalent expressions for $P(1,t|2,t_0)$ and $P(2,t|2,t_0)$:

$$P(1,t|2,t_0)=\frac{\omega_{2\to1}\left(1-\mathrm{e}^{-(\omega_{1\to2}+\omega_{1\to2})(t-t_0)}\right)}{\omega_{2\to1}+\omega_{1\to2}}, \tag{8.16}$$

$$P(2,t|2,t_0)=\frac{\omega_{1\to2}+\omega_{2\to1}\mathrm{e}^{-(\omega_{2\to1}+\omega_{1\to2})(t-t_0)}}{\omega_{2\to1}+\omega_{1\to2}}. \tag{8.17}$$

In terms of these conditional probabilities, we can reason that the probability that the particle is in state 1 at time t is the probability that it was in state 1 at time t_0 times the probability $P(1,t|1,t_0)$ plus the probability that it was in state 2 at t_0 times the probability $P(1,t|2,t_0)$:

$$P_1(t)=P_1(t_0)P(1,t|1,t_0)+P_2(t_0)P(1,t|2,t_0) \tag{8.18}$$

and similarly

$$P_2(t)=P_1(t_0)P(2,t|1,t_0)+P_2(t_0)P(2,t|2,t_0). \tag{8.19}$$

In fact, the conditional probabilities $P(i,t|j,t_0)$ of a Markov process cannot be arbitrary functions. If we consider an intermediate time $t_0<t_1<t$, the probability that the particle is in state 1 at time t, provided it was in state 2 at time t_0, can be computed using the probabilities that the particle was at the intermediate time t_1 in one of the two states. Namely,

$$P(1,t|2,t_0)=P(1,t|1,t_1)P(1,t_1|2,t_0)+P(1,t|2,t_1)P(2,t_1|2,t_0) \tag{8.20}$$

and similar equations for $P(1,t|1,t_0)$, $P(2,t|2,t_0)$, and $P(2,t|1,t_0)$. The first term on the right-hand side is the probability that the particle went from 2 at time t_0

to 1 at time t and passed through 1 at time t_1 and a similar interpretation for the second term. These relations are called the *Chapman–Kolmogorov equations* for this process. If we knew the conditional probabilities $P(i,t|j,t_0)$ for arbitrary times t, t_0, the solution of these two functional equations (8.18 and 8.19) would allow us to compute $P_1(t)$ and $P_2(t)$. However, it is difficult to (i) obtain directly from the rules of the process the conditional probabilities $P(i,t|j,t_0)$, and (ii) to solve a functional equation. We have circumvented these problems in our previous treatment by considering a small time increment $t - t_0 = dt$. In this limit of small dt, we find two advantages: (i) the conditional probabilities can be obtained from the rates, that is, $P(1,t+dt|2,t) = \omega(2\to 1)dt + O(dt^2)$, and so on, and (ii) by expanding $P_1(t+dt) = P_1(t) + \dfrac{dP_1(t)}{dt}dt + O(dt^2)$, we obtain a differential instead of a functional equation.

When we discuss in Chapter 9 the use of numerical algorithms to simulate numerically master equations, we will need to use the probability density $f^{1\text{st}}(j,t|i,t_0)$ of the first jump from $i \to j$. This is defined such that $f^{1\text{st}}(j,t|i,t_0)dt$ is the probability that the particle is at state i at time t_0 and stays there until it jumps to state j in the time interval $(t, t+dt)$, *with no intermediate jumps in the whole interval* (t_0, t). Let us now derive directly this function for the particular case of the two-state system. We divide the time interval (t_0, t) in subintervals of length dt: $(t_0, t_0+dt), (t_0+dt, t_0+2dt), \ldots, (t-dt, t)$. The particle is in, say, state $i=1$ at time t_0 and does not make any jump to state 2 until the time interval $(t, t+dt)$. This means that it does not jump during any of the intervals $(t_0+(k-1)dt, t_0+kdt)$ for $k = 1, \ldots, K = (t-t_0)/dt$. The probability that it does not jump during the interval $(t_0+(k-1)dt, t_0+kdt)$ is $1-\omega(1\to 2)dt$. Therefore, the probability that it does not jump during any of these intervals is the product of all these probabilities, $\prod_{k=1}^{K}(1-\omega(1\to 2)dt) = (1-\omega(1\to 2)dt)^K = (1-\omega(1\to 2)dt)^{(t-t_0)/dt}$. In the limit of $dt \to 0$, this becomes[2] equal to $e^{-\omega(1\to 2)(t-t_0)}$. Finally, we have to multiply by the probability that there is a jump from 1 to 2 during the interval $(t, t+dt)$, which is $\omega(1\to 2)dt$. This yields, finally

$$f^{1\text{st}}(2,t|1,t_0) = \omega(1\to 2)e^{-\omega(1\to 2)(t-t_0)}. \tag{8.21}$$

Similarly, we obtain

$$f^{1\text{st}}(1,t|2,t_0) = \omega(2\to 1)e^{-\omega(2\to 1)(t-t_0)}. \tag{8.22}$$

8.1.2 The Occupation Numbers Point of View

We have considered so far a particle that can jump between two states and asked for the probability for that particle to be in state 1 or in state 2. We now consider a system composed of N such particles, each one of them making the random jumps from state 1 to 2, or vice versa. We introduce the occupation number n of state 1 and ask now for the probability $p(n;t)$ that at a given time t exactly n of the N

2) We use $\lim_{x\to 0}(1+x)^{a/x} = e^a$ with $x = -\omega(1\to 2)dt$ and $a = -\omega(1\to 2)(t-t_0)$.

particles are in that state. If the N particles are independent of each other and they all start in the same state, such that $P_1(t)$ is the same for all particles, then $p(n;t)$ is given by a binomial distribution

$$p(n;t) = \binom{N}{n} P_1(t)^n (1 - P_1(t))^{N-n}. \tag{8.23}$$

We will now find directly a differential equation satisfied by $p(n;t)$. This is not really necessary in this simple case, but the techniques we will develop will be useful in the cases in which the jumps between states do not occur independently for each of the particles, or particles start in different initial conditions.

As before, we will relate the probabilities at times t and $t + dt$. Recall, first, that the total number of particles is N and if there are n particles in state 1, then the remainder $N - n$ are in state 2. How can we have n particles in state 1 at time $t + dt$? There are three possibilities:

1) There were n particles in state 1 and $N - n$ in state 2 at time t and none left the state it was in;
2) There were $n - 1$ particles in state 1, and one of the $N - (n - 1)$ particles that were in state 2 jumped from 2 to 1 during that interval;
3) There were $n + 1$ particles in state 1 and one particle jumped from 1 to 2 during that interval.

As we are considering the limit of a small time interval dt, other possible processes (e.g., the jump of two or more particles from one state to another) need not be included, as their contribution will be of a higher order in dt. So, we write

$$\begin{aligned} p(n;t+dt) = {} & p(n;t) \times \text{Prob(no particle jumped)} \\ & + p(n-1;t) \times \text{Prob(any of the } N-n+1 \text{ particles jumps from } 2 \to 1) \\ & + p(n+1;t) \times \text{Prob(any of the } n+1 \text{ particles jumps from } 1 \to 2) \\ & + O(dt^2). \end{aligned} \tag{8.24}$$

Let us analyze each term one by one:

1) The probability that no particle jumped is the product of probabilities that none of the n particles in state 1 jumped to 2 and none of the $N - n$ particles in state 2 jumped to 1. The probability that one particle does not jump from 1 to 2 is $1 - \omega_{1\to 2} dt$; hence, the probability that none of the n particles jumps from 1 to 2 is the product for all particles of this probability, or $[1 - \omega_{1\to 2} dt]^n$. Expanding to first order in dt, this is equal to $1 - n\omega_{1\to 2} dt + O(dt^2)$. Similarly, the probability that none of the $N - n$ particles in 2 jumps to 1 is $1 - (N - n)\omega_{2\to 1} dt + O(dt^2)$. Finally, the probability that no jump occurs whatsoever is the product of these two quantities, or $1 - (n\omega_{1\to 2} + (N - n)\omega_{2\to 1})dt + O(dt^2)$.
2) The probability that one particle jumps from 2 to 1 is $\omega_{2\to 1} dt$; hence the probability that any of the $N - n + 1$ particles jumps from $2 \to 1$ is the sum of all these probabilities, or $(N - n + 1)\omega_{2\to 1} dt$.
3) The probability that one particle jumps from 1 to 2 is $\omega_{1\to 2} dt$; hence the probability that any of the $n + 1$ particles jumps from $1 \to 2$ is the sum of all these probabilities, or $(n + 1)\omega_{1\to 2} dt$.

Again, in all these expressions there could be higher order terms corresponding to multiple intermediate jumps. Putting all these terms together, we obtain

$$\begin{aligned} p(n;t+dt) = {} & p(n;t)(1-(n\omega_{1\to 2}+(N-n)\omega_{2\to 1})dt) \\ & + p(n-1;t)(N-n+1)\omega_{2\to 1}dt \\ & + p(n+1;t)(n+1)\omega_{1\to 2}dt + O(dt^2). \end{aligned} \tag{8.25}$$

Rearranging and taking the limit $dt \to 0$, we obtain the following differential equations:

$$\begin{aligned} \frac{\partial p(n;t)}{\partial t} = {} & -(n\omega_{1\to 2}+(N-n)\omega_{2\to 1})p(n;t) \\ & + (N-n+1)\omega_{2\to 1}p(n-1;t) + (n+1)\omega_{1\to 2}p(n+1;t). \end{aligned} \tag{8.26}$$

This set of $N+1$ equations for the functions $p(n;t)$, which is valid for $n = 0, 1, \ldots, N$, constitutes the master equation from the occupation number point of view, and is complementary to the one-particle point of view considered in Section 8.1.1.

As this set of $N+1$ coupled differential equations is certainly much more complicated than (8.4 and 8.5), and given that we can reconstruct $p(n;t)$ from $P_1(t)$ using (8.23), one can ask what is the usefulness of such approach. The answer is that (8.23) is only valid in the case of independent particles, an assumption which is not always correct. For example, it is true in the case of the radioactive atoms in which each atom decays independently of others. However, it is not correct in the vast majority of cases of interest. Consider the example in which state 1 is "healthy" and state 2 is "sick" for a contagious disease. The rate $\omega_{1\to 2}$ at which a healthy person gets infected (the rate at which he goes from 1 to 2) depends naturally on the number $N-n$ of sick persons, as the more infected people there are around one person, the higher the probability that this person gets the disease, whereas we might consider that the recovery rate $\omega_{2\to 1}$ is independent on how many people are infected. In a chemical reaction, the rate at which Cl and Na react to form NaCl depends on how many free (not combined) atoms exist, and so on. In all these cases, it is not possible to write down closed equations for the probability that one particle is in state 1 independently of how many particles share this state. When this happens, master equations of the form of (8.26) are the necessary starting point.

The general structure of the master equation obtained here is not so different of the one derived in Section 8.1.1 taking the particle point of view. We can introduce global rates to jump between global states with n particles in state 1. In this case, we define the rates

$$\Omega(n \to n+1) = (N-n)\omega_{2\to 1}, \tag{8.27}$$

$$\Omega(n \to n-1) = n\omega_{1\to 2} \tag{8.28}$$

and write the master equation as

$$\begin{aligned} \frac{\partial p(n;t)}{\partial t} = {} & -(\Omega(n\to n-1)+\Omega(n\to n+1))p(n;t) \\ & + \Omega(n-1\to n)p(n-1;t) + \Omega(n+1\to n)p(n+1;t) \end{aligned} \tag{8.29}$$

which is the same kind of balance relation that we can find in (8.4 and 8.5). The first term on the right-hand side is the loss of probability from n to either $n-1$ or $n+1$, whereas the second and third terms represent the gain of probability from states with $n-1$ or $n+1$ particles, respectively.

Before we generalize these results to particles that can be in more than two states, let us introduce a simplifying notation. We introduce the so-called step operator E. This is a linear operator defined for any function $f(n)$ which depends on an integer variable n, and it is defined as

$$E[f(n)] = f(n+1) \tag{8.30}$$

which allows us to define powers of this operator as

$$E^{\ell}[f(n)] = f(n+\ell) \tag{8.31}$$

for $\ell \in \mathbb{Z}$ an integer number, in particular $E^{-1}[f(n)] = f(n-1)$. Using this notation, (8.26) can be written in the compact form

$$\frac{\partial p(n;t)}{dt} = (E-1)[\Omega(n \to n-1)p(n;t)] + (E^{-1}-1)[\Omega(n \to n+1)p(n;t)]. \tag{8.32}$$

Note that the first term, the one with the operator $(E-1)$, represents the loss of probability of processes in which particles leave state 1 to go to state 2 and hence decrease the population of state 1. On the other hand, the term including the operator $(E^{-1}-1)$ represents those processes that increase the population of 1 by including the contribution of the jumps from 2 to 1. This will be a common feature of more general master equations.

8.2 The General Case

We now briefly introduce the notation and equations in the case where there are more than two states, as the generalization is straightforward. If one particle can be in many, but numerable, discrete states $i = \ldots, -2, -1, 0, 1, 2, \ldots$, we denote by $P_i(t)$ the probability that it is in state i at time t. From this state, it can jump to state j with rate $\omega(i \to j)$. As it can jump, in principle, to any other state, the probability that it leaves state i in the time interval $(t, t+dt)$ is $\sum_{j\neq i} \omega(i \to j)dt$ and the probability that it remains in state i during the same time interval is $1 - \sum_{j\neq i} \omega(i \to j)dt$. At the same time, the probability that there is a jump from any other state j into state i at this time interval is $\sum_{j\neq i} \omega(j \to i)dt$. So we can write for the probability of being in state i at time $t+dt$ as

$$P_i(t+dt) = P_i(t)[1 - \sum_{j\neq i} \omega(i \to j)dt] + \sum_{j\neq i} P_j(t)\omega(j \to i)dt + O(dt^2). \tag{8.33}$$

Again, after rearranging and taking the limit $dt \to 0$, we obtain the *master equation* for a discrete process:

$$\frac{dP_i(t)}{dt} = \sum_{j\neq i} \left[-\omega(i \to j)P_i(t) + \omega(j \to i)P_j(t)\right] \tag{8.34}$$

or, in terms of the currents $J(i \to j) = -\omega(i \to j)P_i(t) + \omega(j \to i)P_j(t)$

$$\frac{dP_i(t)}{dt} = \sum_{j \neq i} J(i \to j). \tag{8.35}$$

Although it is not very common in practice, nothing prevents us from considering the more general case where the transition rates depend on time. Hence, a more general master equation is

$$\frac{dP_i(t)}{dt} = \sum_{j \neq i} \left[-\omega(i \to j; t)P_i(t) + \omega(j \to i; t)P_j(t) \right]. \tag{8.36}$$

To find the solution of this set of equations, we need to specify an initial condition $P_i(t = t_0), \forall i$.

A natural extension is to consider that the states i in which a particle can be form a continuum, instead of a discrete set. We can think, for instance, in the position of a Brownian particle than can jump randomly from point x to point y. In this case, we need to define $f(x; t)$ as the probability density of being in location x at time t. In other words, $f(x; t)dx$ is the probability that one particle is found in the interval $(x, x + dx)$. Similarly, we need to define the transition rate $w(y \to x)$ from y to x such that $w(y \to x)dxdt$ is the probability of jumping to the interval $(x, x + dx)$ during the time interval $(t, t + dt)$ given that it was at the point y at time t. With these definitions, it is not difficult to generalize the master equation as

$$\frac{\partial f(x; t)}{\partial t} = \int dy \, \left[w(y \to x)f(y; t) - w(x \to y)f(x; t) \right] \tag{8.37}$$

which was already mentioned at the end of Chapter 1.

Let us go back to the discrete case. We stress, again, that the transition rates $\omega_{i \to j} \equiv \omega(i \to j)$ do not need to satisfy any relation among them[3]. Remember also that $\omega(i \to j)$ are rates, not probabilities, and besides having units of $[\text{time}]^{-1}$ do not need to be bounded to the interval $[0, 1]$ (although they are nonnegative quantities). It is easy now to verify that whatever the coefficients $\omega(i \to j)$, it follows from (8.36) the conservation of the total probability that

$$\frac{d}{dt} \sum_i P_i(t) = 0 \tag{8.38}$$

and, again, we have the normalization condition $\sum_i P_i(t) = 1$ for all times t provided that $\sum_i P_i(t_0) = 1$.

When the total number of states N is finite, it is possible, and useful sometimes, to define the matrix $\mathbf{W}$ as

$$\begin{array}{lcll} W_{ij} & = & \omega(j \to i) & \text{if } i \neq j, \\ W_{ii} & = & -\sum_{j \neq i} \omega(i \to j). & \end{array} \tag{8.39}$$

3) The elements $\omega(i \to i)$ are not defined and one usually takes $\omega(i \to i) = 0$ although their precise value is irrelevant in most formulas. Note also that the sum in (8.36) can be replaced by $\sum_{\forall j}$ since the term $j = i$ does not contribute to this sum whatever the value of $\omega_{i \to i}$.

$W_i \equiv |W_{ii}|$ is nothing but the total escape rate from the state i as the sum of all rates to all possible states. In terms of the coefficients W_{ij}, the rate equations admit the matrix form

$$\frac{dP_i(t)}{dt} = \sum_j W_{ij} P_j(t). \tag{8.40}$$

The matrix W is such that the rows add up to zero. This property ensures that the solutions $P_i(t)$ respect the positivity condition $P_i(t) \geq 0$ provided that $P_i(t_0) \geq 0$.

We now determine the pdf $f^{1\text{st}}(j, t|i, t_0)$ of the first jump from $i \to j$. The definition is the same as before: $f^{1\text{st}}(j, t|i, t_0)dt$ is the probability that the particle is at state i at time t_0 and stays there until it jumps to state j in the time interval $(t, t + dt)$, *with no intermediate jumps to any other state in the interval* (t_0, t). We need not repeat the proof we gave in the case of two states. As the probability that the system jumps from i to any other state in the time interval $(t, t + dt)$ is $W_i dt$, the probability that there have been no jumps in the interval (t_0, t) is $\mathrm{e}^{-W_i(t-t_0)}$. As the probability that there is a jump from i to j in the time interval $(t, t + dt)$ is $\omega(i \to j)dt$, the required probability density function is

$$f^{1\text{st}}(j, t|i, t_0) = \omega(i \to j)\mathrm{e}^{-W_i(t-t_0)}. \tag{8.41}$$

This will be useful when discussing the numerical methods for simulating a master equation.

We can adopt also the occupation numbers point of view and ask for the probability $p(n_1, n_2, \ldots; t)$ that there are n_1 particles in state 1, n_2 in state 2, and so on. Instead of writing general formulas, we will now consider some specific examples.

8.3 Examples

8.3.1 Radioactive Decay

Let us take as an example a β-radioactive substance. The events are the emission of electrons by an atom at an individual rate ω. Schematically

$$\mathrm{X} \longrightarrow \mathrm{Y} \tag{8.42}$$

where X (state 1) denotes a radioactive atom and Y (state 2) the product of the disintegration. In fact, this example is nothing but a simplification of the two-state case considered in Section 8.1, because the reverse transition does not occur. All we need to do then is to take $\omega(1 \to 2) = \omega$ and $\omega(2 \to 1) = 0$. The initial condition at $t_0 = 0$ is that the atom is in the 1 state, or $P_1(t_0) = 1$. Then, the probability that this atom has not yet disintegrated at time t is

$$P_1(t) = \mathrm{e}^{-\omega t}. \tag{8.43}$$

If at time $t = 0$ there are N atoms that have not yet disintegrated, and as atoms disintegrate independently of each other, the probability that at time t there are n atoms not yet disintegrated is given by the binomial distribution (8.23), or

$$p(n;t) = \binom{N}{n} e^{-n\omega t}(1 - e^{-\omega t})^{N-n}. \tag{8.44}$$

The average value of the binomial distribution is

$$\langle n(t)\rangle = \sum_n np(n;t) = Ne^{-\omega t} \tag{8.45}$$

which is the law of radioactive decay.

The probability $p(n;t)$ satisfies the master equation (8.32) with $\Omega(n \to n+1) = 0$, as it follows from $\omega_{2\to 1} = 0$ consistent with the intuitive condition that there is no mechanism by which the number of nondisintegrated atoms can be increased:

$$\frac{\partial p(n;t)}{\partial t} = (E-1)\left[\Omega(n \to n-1)p(n;t)\right]. \tag{8.46}$$

It is left as an exercise to check that (8.44) indeed satisfies this equation.

8.3.2 Birth (from a Reservoir) and Death Process

An important class of master equations responds to the *birth-and-death* scheme. In one of its simplest forms, we assume that particles are created at a constant rate ω_A out of N_A sources. Once created, these particles can disappear at another constant rate ω. This is schematized as follows:

$$A \longrightarrow X \longrightarrow \emptyset. \tag{8.47}$$

The sources A are assumed to be endless, so that the rate of production of the X particles is always constant, irrespective of how many particles have been already created. The set of N_A sources is the ''reservoir'' out of which the X particles are created.

We take the occupation numbers point of view and focus on the probability $p(n;t)$ that there are n X particles at time t. We have now three elementary contributions to $P(n;t+dt)$ according to what happened in the time interval $(t, t+dt)$:

1) There were n X particles at time t and none was lost or created from the reservoir. The probability that one source A does not create a particle in the interval $(t, t+dt)$ is $1 - \omega_A dt$, and the probability that none of the N_A sources creates a particle is $(1-\omega_A dt)^{N_A} = 1 - \Omega_A dt + O(dt^2)$, with $\Omega_A \equiv N_A\omega_A$. The probability that one of the X particles does not disappear is $1 - \omega dt$, and the probability that none of the n X particles disappears is $(1-\omega dt)^n = 1 - n\omega dt + O(dt^2)$. Hence, the probability that nothing occurs in the interval $(t, t+dt)$ is the product of these two probabilities.
2) There were $n+1$ X particles at time t and one particle disappeared. The probability of one particle disappearing is ωdt, and the probability of any of the $n+1$ particles disappearing is the sum of these probabilities, or $(n+1)\omega dt$.

3) There were $n-1$ X particles and one was created from the reservoir. As each one of the N_A sources has a probability $\omega_A dt$ of creating a particle, the total probability for this event is $N_A\omega_A dt = \Omega_A dt$.

Combining the probabilities of these events, we get

$$p(n;t+dt) = p(n;t)[1-\omega n dt][1-\Omega_A dt] + p(n+1;t)(n+1)\omega dt + p(n-1;t)\Omega_A dt + O(dt^2). \tag{8.48}$$

Rearranging, and taking the limit $dt \to 0$, we get the master equation

$$\frac{\partial p(n;t)}{\partial t} = -(n\omega + \Omega_A)p(n;t) + (n+1)\omega p(n+1;t) + \Omega_A p(n-1;t) \tag{8.49}$$

or in terms of the step operator

$$\frac{\partial p(n;t)}{\partial t} = (E-1)[\Omega(n \to n-1)p(n;t)] + (E^{-1}-1)[\Omega(n \to n+1)p(n;t)] \tag{8.50}$$

with

$$\Omega(n \to n-1) = n\,\omega, \tag{8.51}$$

$$\Omega(n \to n+1) = \Omega_A = N_A\omega_A. \tag{8.52}$$

Again, the term $E-1$ corresponds to the destruction of X particles, whereas the term $E^{-1}-1$ corresponds to their creation. This time, however, there is no *a priori* upper limit for the value of the number of X particles and therefore $n = 0, 1, \ldots, \infty$. The master equation consists in infinite coupled equations that have to be solved using the initial conditions $P(n;0) = \delta_{n,N_0}$, $n = 0, 1, \ldots$, with N_0 being the number of X particles present at the initial time $t = 0$. We will find its solution in a later section.

8.3.3 A Chemical Reaction

We consider the simple chemical reaction in which an atom A and an atom B combine to give the molecule AB. The reaction is reversible, so the molecule AB can break up in the constituent atoms.

$$\text{A+B} \rightleftarrows \text{AB}. \tag{8.53}$$

For simplicity, let us consider a situation in which initially there are the same number N of A atoms and B atoms and no AB molecules. An A atom can be not bound (state 1), or bound to B to form AB (state 2). We denote by $n(t)$ the number of A atoms in state 1 at time t. As one A atom combines with a B atom, the number of B atoms at time t is also $n(t)$, and the number of AB molecules is $N - n(t)$. The combination of an A and a B atom to form a molecule is a complicated process for which we adopt a probabilistic point of view. We assume that the rate at which an individual A atom combines with a B atom to form the product AB is $\omega_{1\to 2}$, while the reverse reaction AB $\to$ A+B happens at a rate $\omega_{2\to 1}$.

We could focus on the particle point of view and write down equations for the probability that one A atom is in state 1 (unbound) or in state 2 (bound). These equations for $P_1(n;t)$ and $P_2(n;t)$ would look similar to (8.4) and (8.5). A new ingredient appears in this case. It is not reasonable to assume that the rate $\omega_{1\to 2}$ is a constant independent on how many B atoms there are. For A to react with B, they first have to meet. We can imagine that the reaction takes place in a container of volume V. If the atoms move freely throughout the whole volume, we can assume that the reaction rate $\omega_{1\to 2}$ is proportional to the particle density n/V of B atoms. We insist that this is an assumption we take to describe the chemical reaction. This assumption will in general not be fully correct as it assumes that the density of B atoms at the neighborhood of an A atom is homogeneous. If we take this assumption, though, we are led to taking the dependence $\omega_{1\to 2}[n] = k_{12} n V^{-1}$, with k_{12} being this time a constant, independent of n and V. On the other hand, it seems reasonable to assume that the breaking up of an AB molecule into its constituent atoms, being an event involving only one molecule, does not depend on the density of atoms or molecules, so the rate $\omega_{2\to 1}$ is independent of n.

To cope with this difficulty, we adopt the occupation numbers point of view and consider the individual events that increase or decrease the number n of A atoms in order to analyze the change in probability during the time interval $(t, t+dt)$. The terms that contribute to $p(n;t+dt)$, which is the probability that at time $t+dt$ there are n A atoms, are as follows:

1) There are n A atoms at time t and none of them reacts with a B atom and none of the $N-n$ molecules breaks up. The probability is $(1-\omega_{1\to 2}[n]dt)^n(1-\omega_{2\to 1}dt)^{N-n} = 1-(n\omega_{1\to 2}[n]+(N-n)\omega_{2\to 1})dt + O(dt^2)$.
2) There are $n+1$ A atoms at time t and one of them reacts with any of the $n+1$ B atoms. The probability that one A atom reacts is $\omega_{1\to 2}[n+1]dt$. The probability that any of the $(n+1)$ A atoms reacts is $(n+1)\omega_{1\to 2}[n+1]dt$.
3) There are $n-1$ A atoms at time t and one of the $N-(n-1)$ AB molecules breaks up into its constituent atoms. This event happens with probability $(N-n+1)\omega_{2\to 1}dt$.

Combining all these events, we get

$$\begin{aligned} p(n;t+dt) = {} & p(n;t)\times(1-(n\omega_{1\to 2}[n]+(N-n)\omega_{2\to 1})dt) \\ & + p(n+1;t)\times(n+1)\omega_{1\to 2}[n+1]dt \\ & + p(n-1;t)\times(N-n+1)\omega_{2\to 1}dt. \end{aligned} \tag{8.54}$$

Rearranging, taking the limit $dt\to 0$, and introducing the step operators, we arrive at

$$\frac{\partial p(n;t)}{dt} = (E-1)[\Omega(n\to n-1)p(n;t)] + (E^{-1}-1)[\Omega(n\to n+1)p(n;t)] \tag{8.55}$$

with

$$\Omega(n\to n-1) = n\omega_{1\to 2}[n] = k_{12}V^{-1}n^2, \tag{8.56}$$

$$\Omega(n\to n+1) = (N-n)\omega_{2\to 1}. \tag{8.57}$$

The fact that the rate $\Omega(n \to n-1) \propto n^2$ is an example of the law of mass action. The process $n \to n-1$ requires that an A atom meets a B atom, an event that is postulated to happen with a probability proportional to the product $n_A n_B$ of the number n_A of A atoms and n_B of B atoms, which (in our simplified treatment) are exactly the same, $n_A = n_B = n$. More generally, if we consider the chemical reaction

$$a\mathrm{A} + b\mathrm{B} \rightleftarrows c\mathrm{C} + d\mathrm{d} \tag{8.58}$$

where a A molecules and b B molecules react to form c C molecules and d D molecules, the global rates are assumed to be

$$\Omega(n_A \to n_A - a, n_B \to n_B - b, n_C \to n_C + c, n_d \to n_d + d) = k n_A^a n_B^b, \tag{8.59}$$

$$\Omega(n_A \to n_A + a, n_B \to n_B + b, n_C \to n_C - c, n_d \to n_d - d) = k' n_C^c n_d^d, \tag{8.60}$$

with k and k' being constants.

8.3.4 Self-Annihilation

In this case, we consider that the X particles are created out of an endless reservoir at a rate ω_A, but disappear in pairs. Schematically,

$$\begin{aligned} A &\to X, \\ X + X &\to \emptyset. \end{aligned} \tag{8.61}$$

We denote by ω the individual rate at which one particle encounters another and leads to the annihilation of both. This example combines ingredients from the two previous examples. While the creation part from the reservoir is identical to the one analyzed in Section 8.3, the same arguments used in the previous chemical reaction case lead us to assume that the individual reaction rate depends on the density of available particles with which a given particle can interact with, or $\omega[n] = kV^{-1}(n-1)$, for a system containing n particles.

To find the master equation for $p(n;t)$ which is the probability of having n X particles at time t, we focus on the elementary processes that can occur between t and $t + dt$:

1) There were n X particles at time t and nothing happened during the time interval $(t, t+dt)$. This includes that no particle was created, with the probability $1 - \Omega_A dt + O(dt^2)$, and no X particles were annihilated, with probability $1 - n\omega[n]dt + O(dt^2)$, for a final probability $1 - (n\omega[n] + \Omega_A)dt + O(dt^2)$.
2) There were $n+2$ particles at time t. Two particles were annihilated with probability $(n+2)\omega[n+2]dt$, corresponding to the product of the number of particles $n+2$ with the rate at which any of them can be annihilated, $\omega[n+2]$.
3) There were $n-1$ X particles and one was created from the reservoir with probability $\Omega_A dt$.

Combining all these terms, we can write

$$\begin{aligned} p(n;t+dt) &= p(n;t)[1-(n\omega[n]+\Omega_A)dt] && \text{case 1} \\ &\quad + p(n+2;t)(n+2)\omega[n+2]dt && \text{case 2} \\ &\quad + p(n-1;t)\Omega_A dt + O(dt^2) && \text{case 3).} \end{aligned} \tag{8.62}$$

Rearranging, taking the limit $dt \to 0$, and introducing the steps operators, it can be written as

$$\frac{\partial p(n;t)}{\partial t} = (E^2-1)[\Omega(n \to n-2)p(n;t)] + (E^{-1}-1)[\Omega(n \to n+1)p(n;t)] \tag{8.63}$$

with

$$\Omega(n \to n-2) = n\omega[n] = kV^{-1}n(n-1), \tag{8.64}$$

$$\Omega(n \to n+1) = \Omega_A \tag{8.65}$$

where the term E^2-1 represents the annihilation of the two particles, and the $E^{-1}-1$ the creation of one particle.

If we now added to the annihilation of particles the death of a single particle at a rate ω', that is, the scheme

$$\begin{aligned} A &\to X, \\ X+X &\to \emptyset, \\ X &\to \emptyset \end{aligned} \tag{8.66}$$

we could do all the detailed algebra again, but the result

$$\begin{aligned} \frac{\partial p(n;t)}{\partial t} &= (E^2-1)[\Omega(n \to n-2)p(n;t)] + (E^{-1}-1)[\Omega(n \to n+1)p(n;t)] \\ &\quad + (E-1)[\Omega(n \to n-1)p(n;t)] \end{aligned} \tag{8.67}$$

with $\Omega(n \to n-1) = n\omega'$ could have been guessed given the interpretation of the terms $E^{\ell}-1$. The reader can fill in the gaps of the proof if he/she finds it necessary.

8.3.5
The Prey–Predator Lotka–Volterra Model

Let us see now an example of the use of master equations in the context of population dynamics. We consider the so-called predator–prey Lotka–Volterra model. In this model, a predator (e.g., a fox) survives and can reproduce thanks to the eating of a prey (e.g., a rabbit) which, in turn, survives and reproduces by eating a natural resource (e.g., grass). There are many simplifications assumed in this model and it can only be considered to be a sketch of the real dynamics. However, this simple modeling is very popular, as it can grasp the essential (but not all) details of the process.

So, we consider an animal species X (the prey), which reproduces by eating an unlimited natural resource, A. The schematic reaction is as follows[4]:

$$A + X \rightarrow 2X, \tag{8.68}$$

4) Note that this assumes some sort of "asexual" reproduction as it is not necessary that two rabbits meet in order to have offspring.

with some rate ω_0. This means that there is a probability $\omega_0 dt$ that a rabbit gives rise to another rabbit at the time interval $(t, t+dt)$. As we have assumed that the resources (grass) are unlimited, the rate ω_0 is considered to be a constant. The population of rabbits at time t is $n_1(t)$. At the same time, the species Y (the predator, the foxes) reproduces by eating species X. Again, schematically

$$X + Y \rightarrow 2Y \tag{8.69}$$

with an individual rate ω_1. This means that there is a probability $\omega_1 dt$ that a fox eats a rabbit and reproduces during the time interval $(t, t+dt)$. As this bears similarities with previous examples, it seems reasonable to assume that this individual rate depends on the density of rabbits present at time t, so we take the dependence[5] $\omega_1[n_1] = k_1 V^{-1} n_1$. Finally, the species Y can die of natural causes at a rate ω_2:

$$Y \rightarrow \emptyset. \tag{8.70}$$

It is possible to add other processes such as the spontaneous death of the prey X, but let us not complicate the model and study the consequences of these simple steps.

We denote by $p(n_1, n_2; t)$ the probability that there are n_1 animals of species X and n_2 animals of species Y at time t. The master equation can be obtained by enumerating the elementary processes occurring in the time interval $(t, t+dt)$ that might contribute to $p(n_1, n_2; t+dt)$, namely

1) The population was (n_1, n_2) at time t and no rabbit reproduced and no rabbit was eaten and no fox died;
2) The population was $(n_1 - 1, n_2)$ at time t and a rabbit reproduced;
3) The population was $(n_1, n_2 + 1)$ at time t and a fox died;
4) The population was $(n_1 + 1, n_2 - 1)$ at time t and a fox ate a rabbit and reproduced.

The contributions to the probability are, respectively

$$\begin{aligned} p(n_1, n_2; t+dt) = {} & p(n_1, n_2; t)[1 - n_1\omega_0 dt][1 - n_2\omega_1[n_1]dt][1 - n_2\omega_2 dt] \\ & + p(n_1 - 1, n_2; t)(n_1 - 1)\omega_0 dt + p(n_1, n_2 + 1; t)(n_2 + 1)\omega_2 dt \\ & + p(n_1 + 1, n_2 - 1; t)(n_2 - 1)\omega_1[n_1 + 1]dt + O(dt^2). \end{aligned} \tag{8.71}$$

Rearranging and taking the limit $dt \rightarrow 0$, we obtain the desired master equation

$$\begin{aligned} \frac{\partial p(n_1, n_2; t)}{\partial t} = {} & -(n_1\omega_0 + n_2\omega_1[n_1] + n_2\omega_2)p(n_1, n_2; t) \\ & + (n_1 - 1)\omega_0 p(n_1 - 1, n_2; t) + (n_2 + 1)\omega_2 p(n_1, n_2 + 1; t) \\ & + (n_2 - 1)\omega_1[n_1 + 1]p(n_1 + 1, n_2 - 1; t). \end{aligned} \tag{8.72}$$

It can also be written using the step operators E_1 and E_2 acting on variables n_1 and n_2, respectively, as

5) As foxes and rabbits live in a two-dimensional space, V has to be considered as a measure of the area, rather than the volume, where they live.

$$\begin{aligned}\frac{\partial p(n_1,n_2;t)}{\partial t} &= (E_1^{-1}-1)[\Omega\left((n_1,n_2)\to(n_1+1,n_2)\right)p(n_1,n_2;t)\\ &+(E_2-1)[\Omega\left((n_1,n_2)\to(n_1,n_2-1)\right)p(n_1,n_2;t),\\ &+(E_1E_2^{-1}-1)[\Omega\left((n_1,n_2)\to(n_1-1,n_2+1)\right)p(n_1,n_2;t)]\end{aligned} \tag{8.73}$$

with

$$\Omega\left((n_1,n_2)\to(n_1+1,n_2)\right)=n_1\omega_0, \tag{8.74}$$

$$\Omega\left((n_1,n_2)\to(n_1,n_2-1)\right)=n_2\omega_2, \tag{8.75}$$

$$\Omega\left((n_1,n_2)\to(n_1-1,n_2+1)\right)=n_2\omega_1[n_1]=k_1V^{-1}n_1n_2. \tag{8.76}$$

The term $(E_1^{-1}-1)$ represents the creation of an X particle, (E_2-1) the annihilation of an Y particle, and $(E_1E_2^{-1}-1)$ the simultaneous annihilation of an X particle and the creation of a Y particle.

It should be clear by now what the general structure of a master equation is, and after some practice the reader should be able to write down the final expression in terms of the step operators without the need to go through all the detailed steps of the proof.

Now that we have learnt how to derive master equations, we have to solve them. This is not an easy task, and that is the main reason why numerical methods to simulate a master equation are so widespread. We will review them in Chapter 9. Before, however, we will see a powerful technique and some approximated methods of solution.

8.4 The Generating Function Method for Solving Master Equations

We now introduce an analytical method to solve master equations. We will start by the example of the birth-and-death process and consider the set of equation (8.32). We define the *generating function* $G(s,t)$ by means of

$$G(s,t)=\sum_{n=-\infty}^{\infty}s^n p(n;t). \tag{8.77}$$

Note that the sum runs over all values of n. This is a technical point which is not always necessary but simplifies the derivation of the solution. We note that, although (8.32) have been derived for a situation in which the variable n can only take values between 0 and N, we can consider them valid for all integer values of n. All we need to do is to set the initial condition such that $p(n;0)=0$ for $n\notin[0,N]$. One can check that $\dfrac{\partial p(n;t)}{\partial t}=0$ for $n\notin[0,N]$, and hence it is $p(n;t)=0$ for $n\notin[0,N]$.

If we know the generating function, we can expand it in series and, using (8.77), identify the coefficients of the series expansion with the probabilities $p(n;t)$. Note

the property

$$G(1,t) = \sum_{n=-\infty}^{\infty} p(n;t) = 1 \tag{8.78}$$

coming from the normalization condition and valid at all times t. Moreover, the knowledge of the generating function allows the easy determination of the moments of the random variable n. For the first moment, we use the trick $n = \frac{\partial s^n}{\partial s}\Big|_{s=1}$:

$$\begin{aligned}\langle n(t)\rangle &= \sum_n np(n;t) = \sum_n \frac{\partial s^n}{\partial s}\bigg|_{s=1} p(n;t) = \frac{\partial \sum_n s^n p(n;t)}{\partial s}\bigg|_{s=1} \\ &= \frac{\partial G(s,t)}{\partial s}\bigg|_{s=1}.\end{aligned} \tag{8.79}$$

For the second moment, we use $n^2 = \frac{\partial}{\partial s}\left(s\frac{\partial s^n}{\partial s}\right)\Big|_{s=1}$ to obtain

$$\langle n^2(t)\rangle = \sum_n n^2 p(n;t) = \frac{\partial}{\partial s}\left(s\frac{\partial G(s,t)}{\partial s}\right)\bigg|_{s=1}. \tag{8.80}$$

We next find a differential equation for $G(s,t)$. We begin by taking the time derivative of (8.77) and replacing (8.32) and the rates (8.27) and (8.28):

$$\begin{aligned}\frac{\partial G(s,t)}{\partial t} &= \sum_{n=-\infty}^{\infty} s^n \frac{\partial p(n;t)}{\partial t} \\ &= \sum_{n=-\infty}^{\infty} s^n \left[(E-1)[n\omega_{1\to 2}p(n;t)] + (E^{-1}-1)[(N-n)\omega_{2\to 1}p(n;t)]\right].\end{aligned} \tag{8.81}$$

Now we use the following result, valid for $k \in \mathbb{Z}$:

$$\begin{aligned}\sum_n s^n(E^k-1)[f(n)] &= \sum_n s^n(f(n+k)-f(n)) \\ &= \sum_n s^n f(n+k) - \sum_n s^n f(n) \\ &= s^{-k}\sum_n s^{n+k} f(n+k) - \sum_n s^n f(n) \\ &= (s^{-k}-1)\sum_n s^n f(n)\end{aligned} \tag{8.82}$$

where in the third line we have used the change of variables $n+k \to n$.

We also use the result

$$\sum_n s^n np(n;t) = \sum_n s\frac{\partial s^n}{\partial s}p(n;t) = s\frac{\partial\left(\sum_n s^n p(n;t)\right)}{\partial s} = s\frac{\partial G(s,t)}{\partial s}. \tag{8.83}$$

Substitution of (8.82) and (8.83) in (8.81) leads, after some simple algebra, to

$$\frac{\partial G(s,t)}{\partial t} = (1-s)\left[(\omega_{1\to 2} + \omega_{2\to 1}s)\frac{\partial G(s,t)}{\partial s} - \omega_{2\to 1}NG(s,t)\right]. \tag{8.84}$$

This is a partial differential equation for the generating function $G(s,t)$. We have hence reduced the problem of solving a set of $N+1$ ordinary differential equations

(8.29) to that of solving one partial differential equation subject to the initial condition $G(s, t = 0) = G_0(s) \equiv \sum_n s^n p(n; 0)$. The solution of (8.84) can be found by the method of the characteristics, and the reader is referred in this point to the wide bibliography in this vast area (see Exercise 4).

We take a simpler and limited approach here. Imagine we want to study just the stationary distribution in which the probabilities $p_{\text{st}}(n)$ are no longer a function of time and the generating function is $G_{\text{st}}(s) = \sum_n s^n p_{\text{st}}(n) = \lim_{t\to\infty} G(s, t)$. This function, being independent of time, satisfies (8.84) with the time derivative equal to zero or, after simplifying the $(1 - s)$ factor

$$0 = (\omega_{1\to 2} + \omega_{2\to 1} s)\frac{dG_{\text{st}}(s)}{ds} - \omega_{2\to 1} N G_{\text{st}}(s). \tag{8.85}$$

This is an ordinary differential equation whose solution under the initial condition following from (8.78), $G_{\text{st}}(1) = 1$ is

$$G_{\text{st}}(s) = \left[\frac{\omega_{1\to 2} + \omega_{2\to 1} s}{\omega_{1\to 2} + \omega_{2\to 1}}\right]^N. \tag{8.86}$$

All that remains is to expand this in powers of s using Newton's binomial theorem.

$$G_{\text{st}}(s) = \sum_{n=0}^{N} \binom{N}{n} \left(\frac{\omega_{2\to 1} s}{\omega_{1\to 2} + \omega_{2\to 1}}\right)^n \left(\frac{\omega_{1\to 2}}{\omega_{1\to 2} + \omega_{2\to 1}}\right)^{N-n} \tag{8.87}$$

which gives

$$p_{\text{st}}(n) = \binom{N}{n} \left(\frac{\omega_{2\to 1}}{\omega_{1\to 2} + \omega_{2\to 1}}\right)^n \left(\frac{\omega_{1\to 2}}{\omega_{1\to 2} + \omega_{2\to 1}}\right)^{N-n} \tag{8.88}$$

which is a binomial distribution of parameter $P_1^{\text{st}} = \dfrac{\omega_{2\to 1}}{\omega_{1\to 2} + \omega_{2\to 1}}$, in full agreement with (8.23) in the steady state using (8.11).

Let us apply, as a last example, the technique of the generating function to the birth-and-death process in Section 8.3. The resulting partial differential equation is

$$\frac{\partial G(s, t)}{\partial t} = (s - 1)\left[\Omega_A G(s, t) - \omega \frac{\partial G(s, t)}{\partial s}\right]. \tag{8.89}$$

It is possible to find the solution of this equation using the method of characteristics. For simplicity, let us focus again in the stationary state in which the time derivative is equal to zero:

$$0 = \Omega_A G_{\text{st}}(s) - \omega \frac{dG_{\text{st}}(s)}{ds}. \tag{8.90}$$

The solution with the initial condition $G_{\text{st}}(1) = 1$ is

$$G_{\text{st}}(s) = \mathrm{e}^{\frac{\Omega_A}{\omega}(s-1)}. \tag{8.91}$$

If we expand it in a power series of its argument

$$G_{\text{st}}(s) = \mathrm{e}^{-\frac{\Omega_A}{\omega}} \sum_{n=0}^{\infty} \left(\frac{\Omega_A}{\omega}\right)^n \frac{s^n}{n!} \tag{8.92}$$

we find

$$p_{\rm st}(n) = \frac{\mathrm{e}^{-\frac{\Omega_A}{\omega}}}{n!}\left(\frac{\Omega_A}{\omega}\right)^n, \tag{8.93}$$

which is a Poisson distribution of the parameter $\lambda = \dfrac{\Omega_A}{\omega}$.

8.5 The Mean-Field Theory

Sometimes (more often than desired), it is not possible to solve the master equation using the generating function technique or any other. However, it is possible to obtain very easily approximated equations for the first moments of the probability $p(n;t)$. In many occasions, the knowledge of the first moment $\langle n(t)\rangle$ gives important information about the underlying stochastic process.

Let us then consider the general master equation

$$\frac{\partial p(n;t)}{\partial t} = \sum_{\ell}(E^{\ell}-1)\left[\Omega_{n\to n-\ell}\,p(n;t)\right]. \tag{8.94}$$

Multiplying by n and summing over n, one gets after some algebra the (exact) equations for the first moment, as

$$\frac{d\langle n(t)\rangle}{dt} = -\sum_{\ell}\left\langle \ell\,\Omega_{n\to n-\ell}\right\rangle. \tag{8.95}$$

Let us apply this result to the radioactive decay discussed in 8.3. The only contribution to the master equation comes from the term $\ell = 1$ with $\Omega(n \to n-1) = \omega n$. Therefore

$$\frac{d\langle n(t)\rangle}{dt} = -\langle \omega n\rangle = -\omega\langle n\rangle. \tag{8.96}$$

This is a closed equation whose solution $\langle n(t)\rangle = N\mathrm{e}^{-\omega t}$ agrees, of course, with (8.45), obtained by other methods.

Other times we are not so lucky. If we take the master equation of the self-annihilation process of 8.3, as given in (8.63), the contributions come from the terms $\ell = 2$ and $\ell = -1$, with the rates (8.64) and (8.65). This gives

$$\frac{d\langle n(t)\rangle}{dt} = -\langle 2kV^{-1}n(n-1)\rangle + \langle \Omega_A\rangle \tag{8.97}$$

$$= -2kV^{-1}\langle n^2\rangle + 2kV^{-1}\langle n\rangle + \Omega_A. \tag{8.98}$$

But, alas! this is a not closed equation, as the evolution of the first moment $\langle n\rangle$ depends on the second moment $\langle n^2\rangle$. It is possible now to derive an equation for the evolution of the second moment using the general result

$$\frac{d\langle n^2(t)\rangle}{dt} = -\sum_{\ell}\left\langle \ell(\ell-2n)\Omega_{n\to n-\ell}\right\rangle. \tag{8.99}$$

However, when we replace the rates (8.64) and (8.65), the evolution of the second moment depends on the third moment, and so on, in an infinite hierarchy of equations.

The simplest simplification scheme to break the hierarchy is to assume that the stochastic process is such that the fluctuations of the n variable can be neglected, or $\sigma^2[n] = \langle n^2\rangle - \langle n\rangle^2 \approx 0$, which implies $\langle n^2\rangle \approx \langle n\rangle^2$. This is the mean-field approximation[6]. Replacing in (8.98), we obtain the closed equation

$$\frac{d\langle n(t)\rangle}{dt} = -2kV^{-1}\langle n\rangle^2 + 2kV^{-1}\langle n\rangle + \Omega_A. \tag{8.100}$$

It is convenient to consider the average density of the X particles, defined as $x(t) = V^{-1}\langle n(t)\rangle$. A simple manipulation leads to

$$\frac{dx(t)}{dt} = -2kx^2 + 2kV^{-1}x + V^{-1}\Omega_A. \tag{8.101}$$

The last term $V^{-1}\Omega_A = V^{-1}\omega_A N_A = \omega_A x_A$, with $x_A = V^{-1}N_A$ being the density of the reservoir. In the thermodynamic limit, $V \to \infty$, we can neglect the second term on the right-hand side and arrive at the macroscopic equation[7]

$$\frac{dx(t)}{dt} = -2kx^2 + \omega_A x_A. \tag{8.102}$$

Let us turn now to the prey–predator Lotka–Volterra model. We begin by computing the average values of the number of prey and predators, $\langle n_1(t)\rangle = \sum_{n_1,n_2} n_1 p(n_1,n_2;t)$ and $\langle n_2(t)\rangle = \sum_{n_1,n_2} n_2 p(n_1,n_2;t)$. Taking the time derivative and replacing the master equation (8.73) and the rates (8.74)–(8.76), one obtains after some algebra

$$\frac{d\langle n_1(t)\rangle}{dt} = \omega_0\langle n_1\rangle - k_1 V^{-1}\langle n_1 n_2\rangle. \tag{8.103}$$

$$\frac{d\langle n_2(t)\rangle}{dt} = k_1 V^{-1}\langle n_1 n_2\rangle - \omega_2\langle n_2\rangle. \tag{8.104}$$

These equations are again not closed. We could now compute the time evolution of $\langle n_1 n_2\rangle = \sum_{n_1,n_2} n_1 n_2 p(n_1,n_2;t)$, but then it would be coupled to higher and higher order moments, a complete mess! We use again the mean-field approach and assume that the correlations between the populations of prey and predator can be neglected, or $\langle n_1 n_2\rangle \approx \langle n_1\rangle\langle n_2\rangle$. Under this approximation, we obtain the closed equations

$$\frac{d\langle n_1(t)\rangle}{dt} = \omega_0\langle n_1\rangle - k_1 V^{-1}\langle n_1\rangle\langle n_2\rangle, \tag{8.105}$$

$$\frac{d\langle n_2(t)\rangle}{dt} = k_1 V^{-1}\langle n_1\rangle\langle n_2\rangle - \omega_2\langle n_2\rangle \tag{8.106}$$

6) The word "mean-field" has different meanings in different contexts. Here, it refers specifically to the assumption that the fluctuations can be neglected.

7) The explicit solution is $x(t) = x_{\rm st}\tanh\left(t/\tau + \operatorname{arctanh}(x(0)/x_{\rm st})\right)$ with $x_{\rm st} = \sqrt{\omega_A x_A/2k}$ and $\tau = 1/\sqrt{2k\omega_A x_A}$.

which can be written in terms of the density of the different species $x_1(t) = V^{-1}\langle n_1(t)\rangle$, $x_2(t) = V^{-1}\langle n_2(t)\rangle$, as

$$\frac{dx_1(t)}{dt} = \omega_0 x_1 - k_1 x_1 x_2, \tag{8.107}$$

$$\frac{dx_2(t)}{dt} = k_1 x_1 x_2 - \omega_2 x_2. \tag{8.108}$$

These are the celebrated Lotka–Volterra equations.

8.6 The Fokker–Planck Equation

The master equation is a complicated set of many coupled differential equations. We have already seen that it can be analyzed in terms of a partial differential equation for the generating function $G(s,t)$. We will now find an approximated partial differential equation valid directly for the probability $p(n;t)$. It is possible to develop a rigorous derivation of this equation by estimating the order of the approximation that occurs in the truncation. Here, we offer a very simplified derivation in which it is not possible to determine precisely the error of the approximation. We limit ourselves to master equations for one variable of the form (8.94), but similar expansions can be carried out in the case of having more than one variable. The idea is to consider that n is a large macroscopic variable and can be treated as continuous[8]. With this mind, we use a Taylor series expansion in the definition of the step operator applied to any function $f(n)$:

$$E^{\ell}[f(n)] = f(n+\ell) = f(n) + \ell\frac{df(n)}{dn} + \frac{\ell^2}{2!}\frac{d^2f(n)}{dn^2} + \dots, \tag{8.109}$$

and

$$(E^{\ell} - 1)[f(n)] = E^{\ell}[f(n)] - f(n) = \ell\frac{df(n)}{dn} + \frac{\ell^2}{2!}\frac{d^2f(n)}{dn^2} + \dots \tag{8.110}$$

where (without much justification) we restrict the expansion to the second order in ℓ. The lack of justification of this truncation is one of the weak points of this simple derivation[9]. Replacing in (8.94), we obtain

$$\frac{\partial p(n;t)}{\partial t} = \sum_{\ell}\left(\ell\frac{\partial(\Omega_{n\to n-\ell}p(n;t))}{\partial n} + \frac{\ell^2}{2!}\frac{\partial^2(\Omega_{n\to n-\ell}p(n;t))}{\partial n^2}\right) \tag{8.111}$$

or, rearranging the terms

$$\frac{\partial p(n;t)}{\partial t} = \frac{\partial}{\partial n}\left(F(n)p(n;t) + \frac{1}{2}\frac{\partial}{\partial n}\left(G(n)p(n;t)\right)\right) \tag{8.112}$$

8) It is possible to formalize this idea by introducing $x = n/V$, with V being a large parameter (not necessarily related to the volume V). The expansion becomes then a power series in V.

9) An interesting result is Pawula's theorem, which states, basically, that only by truncating at second order can the resulting equation (8.112) be ensured to have the property that the probability $p(n;t)$ remains positive at all times.

with

$$F(n) = \sum_{\ell} \ell\, \Omega_{n\to n-\ell}, \tag{8.113}$$

$$G(n) = \sum_{\ell} \ell^2 \Omega_{n\to n-\ell}. \tag{8.114}$$

Equation (8.112) has the form of a Fokker–Planck equation for the probability $p(n;t)$ with $F(n)$ and $G(n)$ the drift and diffusion coefficients, which we have already encountered in the previous chapters. Just to finish this chapter, we mention that, using the result of exercise (6.10) one can write down the associated Langevin equation (in the Itô interpretation):

$$\frac{dn(t)}{dt} = -F(n) + \sqrt{G(n)}\xi(t) \tag{8.115}$$

where $\xi(t)$ is the usual zero-mean delta-correlated white noise.

Further Reading and References

The theory of master equations can be found, explained to a much deeper level than the one used here, in the book by van Kampen [24]. Pawula's theorem appears in [40]. The classic Lotka–Volterra equations and their solutions are explained in many books; see, for example, [41].

Exercises

1) Check that the conditional probabilities (8.14–8.17) satisfy the Chapman–Kolmogorov equations (8.20) and that the probabilities 8.9 and 8.10 satisfy the functional relations 8.18 and 8.19).
2) Derive the pdf of the first jump in the case of time-dependent rates $\omega_{ij}(t)$ as

   $$f^{1\text{st}}(j,t|i,t_0) = \omega_{ij}(t)\mathrm{e}^{-\int_{t_0}^{t} ds\, W_{ii}(s)}$$

   with $W_{ii}(t) = \sum_{j\neq i} \omega_{ij}$.
3) Consider the general master equation

   $$\frac{\partial P(n,t)}{\partial t} = \sum_{\ell} (E^{\ell} - 1)\left[C_{\ell}(n)\, P(n,t)\right]$$

   where we use the notation $C_{\ell}(n) = \Omega_{n,n-\ell}$. Assume that the transition rates can be expanded in a Taylor series $C_{\ell}(n) = \sum_k C_{\ell}^k n^k$, and use that $s^{n+k} = (s\frac{\partial}{\partial s})^k s^n$ to derive the equation for the generating function

   $$\frac{\partial G}{\partial t} = \sum_{\ell} (s^{-\ell} - 1) C_{\ell}\left(s\frac{\partial}{\partial s}\right) G(s,t).$$

4) Check by direct substitution that

   $$G(s,t) = \left[\frac{\omega_1 + \omega_2 s + \omega_2 e^{-\omega t}(1-s)}{\omega}\right]^N G_0\left(\frac{\omega_1 + \omega_2 s - \omega_1 e^{-\omega t}(1-s)}{\omega_1 + \omega_2 s + \omega_2 e^{-\omega t}(1-s)}\right)$$

with $\omega \equiv \omega_1 + \omega_2$, is the general solution of (8.84). Set as initial conditions (i) $p(n; t = 0) = \delta_{n,0}$ and (ii) $p(n; t = 0) = \delta_{n,N}$, and determine the function G_0 and check that in both cases $p(n; t)$ is given by a binomial distribution.

5) Check by direct substitution that

$$G(s,t) = \mathrm{e}^{\frac{\Omega_A}{\omega}(s-1)(1-\mathrm{e}^{-\omega t})} G_0\left(1 + (s-1)\mathrm{e}^{-\omega t}\right)$$

is the general solution of (8.32). Use the derivatives of this function to find the first two moments $\langle n(t)\rangle$ and the variance $\sigma^2[n(t)]$ that follows from the master equation.

6) Derive the mean-field equation

$$\dot{x}(t) = \omega_{21}(v^{-1} - x) - k_{12}x^2$$

for the chemical reaction indicated in Section 8.3. Here, $v = V/N$ is the volume per atom and $x = n/V$ is the density of NaCl molecules.

7) Derive the Lotka–Volterra mean-field equations in the case where we consider separately and with different rates the processes $X + Y \longrightarrow 2Y$ and $X + Y \longrightarrow Y$. Show that these general equations

$$\frac{dx_1(t)}{dt} = \omega_0 x_1 - k_1 x_1 x_2,$$
$$\frac{dx_2(t)}{dt} = k_2 x_1 x_2 - \omega_2 x_2$$

admit the constant of movement $x_1^{\omega_2} x_2^{\omega_0} \mathrm{e}^{-k_1 x_2 - k_2 x_1}$. Conclude that the trajectories $x_1(t)$, $x_2(t)$ are periodic and happen on the orbit

$$x_2 = -\frac{\omega_0}{k_1} W\left[-\frac{k_1}{\omega_0} x_2^0 \mathrm{e}^{-\frac{k_1}{\omega_0}x_2^0} \left(\frac{x_1^0}{x_1}\right)^{\omega_2/\omega_0} \mathrm{e}^{\frac{k_2}{\omega_2}(x_1 - x_1^0)}\right]$$

depending on the initial condition (x_1^0, x_2^0), and the convenient branch of the Lambert function $W[x]$ has to be used as necessary to ensure continuity.

8) The enzymatic reaction. L. Michaelis and M.L. Menten analyzed the enzymatic reaction in which a ''substance'' S becomes a ''product'' with the help of an enzyme E. The enzyme works by binding to the product and facilitating the transformation of S into P, but E it is left unchanged afterward. Schematically,

$$\mathrm{S{+}E} \underset{k_{-1}}{\overset{k_1}{\rightleftarrows}} \mathrm{ES} \underset{k_{-2}}{\overset{k_2}{\rightleftarrows}} \mathrm{P{+}E}$$

where ES is an intermediate molecule and we have indicated by k_i the corresponding rates. By assuming a high energy barrier for the combination of a product with an enzyme, the backward rate k_{-2} can be neglected. Write down the mean-field equations for the evolution of the concentration of substance $s(t)$, enzyme $e(t)$, intermediate molecule $c(t)$, and product $p(t)$. Study the steady-state limit in which the intermediate molecule concentration is a constant, $\dot{c}(t) = 0$.

9) A simple model for the spread of an epidemic is the so-called SIR model: S (for susceptible), I (for infectious), and R (for recovered). In its simplest form, a population of N individuals is split into these three groups: n_S susceptible people can get the disease; n_I infectious people have the disease and can hence pass the infection to susceptible people; and n_R people recovered from the infection who then become immune to new contagion. In this simple version, there is no death or birth of individuals and the total number $N = n_R + n_I + n_R$ remains constant. The process is schematized as

$$\mathrm{S} + \mathrm{I} \overset{\beta}{\longrightarrow} 2\mathrm{I},$$
$$\mathrm{I} \overset{\nu}{\longrightarrow} \mathrm{R}$$

with individual rates β and ν. Write down the mean-field equations for this model and solve them numerically. Assume that the total contagion rate is $\beta n_S n_I / V$, where V is a measure of the volume available to the population.

10) Derive, using the method explained in the text, the Langevin equation corresponding to the birth-and-death process of Section 8.3. Solve it numerically and check the precision with which the Poisson distribution is obtained in the steady state.

11) Write down the master equation corresponding to the SIR model of the previous problem using as independent variables n_S and n_I, that is, the number of susceptible and infected people. Expand the master equation up to second-order derivatives and obtain the 2-D Fokker–Planck equation and, from there, the Langevin equation.

12) Find the general expression for the stationary solution of the Fokker–Planck equation (8.112). Apply the result to the birth-and-death process with rates (8.51 and 8.52) and compare the result with the exact stationary distribution given by (8.93).

9
Numerical Simulations of Master Equations

9.1
The First Reaction Method

Given the difficulties one encounters for the analytical treatment of master equations, it is common to resort to numerical simulations of the underlying stochastic process. We will introduce in this chapter the basic principles of such a procedure. We first begin by the simple example of a two-state system with constant rates. If we denote by X and Y the possible states, there will be jumps from X to Y at a rate $\omega_{1\to 2}$ and from Y to X at a rate $\omega_{2\to 1}$. Remember that, besides being both nonnegative numbers, the rates $\omega_{1\to 2}$ and $\omega_{2\to 1}$ need not have any relation between them. The process has been schematized in (8.1).

The stochastic process consists in a series of jumps from one of the two states to the other. Our numerical simulation in this simple case will consist precisely in the generation of these trajectories that jump from X to Y at the adequate times. Imagine that, at the initial time t_0, we are in the state X. We now need to find the time t_1 at which the particle will jump to Y for the first time. We have already proven in Section 8.1 that this time t_1 of the first jump follows the distribution (8.21):

$$f^{1\text{st}}(2, t_1|1, t_0) = \omega_{1\to 2} \mathrm{e}^{-\omega_{1\to 2}(t_1 - t_0)}. \tag{9.1}$$

In other words, $t_1 - t_0$ follows an exponential distribution. We learnt in Section 2.4 how to generate values of a random variable distributed according to an exponential distribution: simply generate a $\hat{\mathbf{U}}(0, 1)$ random number u_0 uniformly distributed in the interval $(0, 1)$ and use $t_1 - t_0 = -\frac{\ln u_0}{\omega_{1\to 2}}$, or

$$t_1 = t_0 - \frac{\ln u_0}{\omega_{1\to 2}}. \tag{9.2}$$

Now we are in state Y and want to determine the time t_2 of the next jump to state X. The time t_2 of the first jump follows the distribution (8.22):

$$f^{1\text{st}}(1, t_2|2, t_1) = \omega_{2\to 1} \mathrm{e}^{-\omega_{2\to 1}(t_2 - t_1)}. \tag{9.3}$$

Therefore, we can generate t_2 by drawing a $\hat{\mathbf{U}}(0, 1)$ random number u_1 from a uniform distribution in $(0, 1)$ and obtain

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

$$t_2 = t_1 - \frac{\ln u_1}{\omega_{2\to 1}}. \tag{9.4}$$

Now we are again in state X. To find the time t_3 of the next jump, we use the transition rate $\omega_{1\to 2}$ and set

$$t_3 = t_2 - \frac{\ln u_0}{\omega_{1\to 2}} \tag{9.5}$$

and so on, alternating between the rates $\omega_{1\to 2}$ and $\omega_{2\to 1}$. In this way, we can simulate a stochastic trajectory according to the rules of the rates of the process. The next program listing can be used to generate the trajectories according to this basic algorithm up to a maximum time `tmax`. The program sets the initial state, 1 or 2, randomly with probability 1/2, the call to the function `i_ran(2)`.

```
program rate2
  implicit double precision(a-h,o-z)
  tmax=50.0d0
  t=0.0d0
  w12=0.5d0
  w21=1.0d0
  i=i_ran(2)
  write(66,*) t,i
  do while (t < tmax)
    if (i == 1) then
      tn=-dlog(ran_u())/w12
      in=2
    else
      tn=-dlog(ran_u())/w21
      in=1
    endif
    t=t+tn
    write(66,*) t,i
    i=in
    write(66,*) t,i
  enddo
end program rate2
```

The program output (in file 66) is prepared such that a direct plot of this file will show the random trajectories. In Figure 9.1, we plot a sample output of this program.

We now consider the more general case in which there are M possible individual states labeled $1, 2, \ldots, M$. Imagine that at time t_0 we are in state i_0. Now, there can be jumps to $M-1$ different states with rates $\omega_{i_0\to j}$ for $j = 1, \ldots, M$, $j \neq i_0$. If $\omega_{i_0\to j} = 0$, then the corresponding jump $i_0 \to j$ is not permitted. The idea is to generate $M-1$ independent $\hat{\mathbf{U}}(0,1)$ random numbers u_0^j for $j = 1, \ldots, M$, $j \neq i_0$, and compute the jumping times to every one of these states as

$$t_{i_0\to j} = \frac{-\ln u_0^j}{\omega_{i_0\to j}}, \quad j = 1, \ldots, M, \ \ j \neq i_0. \tag{9.6}$$

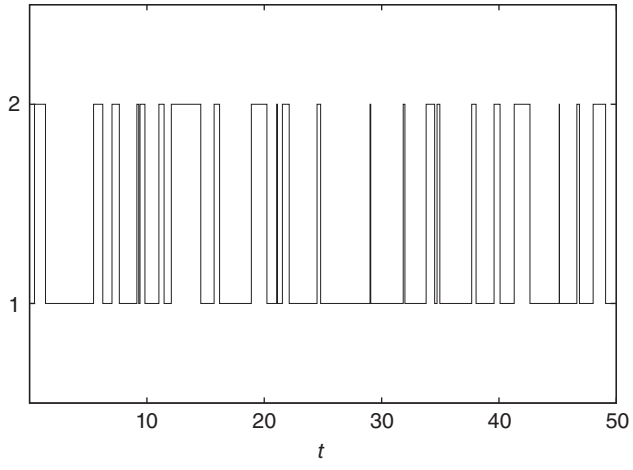


Figure 9.1 Typical trajectory of a stochastic system that jumps between two states using the rates $\omega_{1\to 2} = 0.5$, $\omega_{2\to 1} = 1.0$.

$t_{i_0\to j}$ is the time at which the transition $i_0 \to j$ would occur if there were no other states to which to jump.[1] Which transition would actually occur first? Simple: the one with the smallest time $t_{i_0\to j}$. This is called the "first reaction method." Let i_1 be the state with the minimum transition time: $t_{i_0\to i_1} = \min(t_{i_0\to 1}, t_{i_0\to 2}, \ldots, t_{i_0\to M})$. Then, at time $t_1 = t_0 + t_{i_0\to i_1}$, we jump from i_0 to i_1. Now that we are at state i_1, we have to determine the state and the time of the next jump. For that, we generate $M-1$ independent $\hat{\mathbf{U}}(0,1)$ random numbers u_1^j for $j = 1, \ldots, M, j \neq i_1$ and compute the times of possible jumps

$$t_{i_1\to j} = \frac{-\ln u_1^j}{\omega_{i_1\to j}}, \quad j = 1, \ldots, M, \ j \neq i_1. \tag{9.7}$$

The actual jump $i_1 \to i_2$ is the one that occurs at the earliest of those times: $t_{i_1\to i_2} = \min(t_{i_1\to 1}, t_{i_1\to 2}, \ldots, t_{i_1\to M})$. Then, at time $t_2 = t_1 + t_{i_1\to i_2}$, the state jumps from i_1 to i_2. The process starts again at state i_2 at time t_2.

Here is a program listing that implements this numerical method. We have decided to use as an example the rates $\omega(i \to j) = |i - j|$, which fulfill the necessary condition $\omega(i \to i) = 0$. The rates are now stored in the vector `w(M,M)`.

```
program ratem
  implicit double precision(a-h,o-z)
  parameter (M=1000)
  dimension w(M,M)
  do i=1,M
```

1) Note that if $\omega_{i_0\to j} = 0$, the corresponding jumping time is $t_{i_0\to j} = \infty$, which is another way of saying that the jumping from i_0 to that particular value of j will never occur. In the programming of this algorithm, one needs to avoid the calculation of jumping times for the transitions that are known not to occur; otherwise, annoying overflow errors will appear.

```
    do j=1,M
      w(i,j)=abs(i-j)
    enddo
  enddo
  tmax=10.0d0

  i=i_ran(M)
  t=0.0d0
  write(66,*) t,i
  do while (t < tmax)
    tn=1.0d16
      do j=1,M
        if (w(i,j) > 1.0d-8) then
           t1=-dlog(ran_u())/w(i,j)
           if (t1 < tn) then
             tn=t1
             in=j
           endif
         endif
       enddo
       t=t+tn
      write(66,*) t,i
      i=in
      write(66,*) t,i
   enddo

end program ratem
```

In this program, in order to avoid using rates $\omega(i \to j)$ which are equal to 0, the time of the next jump is not computed if $\omega(i \to j) < 10^{-8}$ (a reasonably small number that might need to be checked in other cases) and we set first a large time `tn`$= 10^{16}$ to determine the minimum jumping time in the assumption that it will be smaller than this big amount (again a reasonable assumption whose validity might need to be checked in one particular application). A simple output of this program for $M = 10$ is plotted in Figure 9.2

Let us now go back to the case of $M = 2$ states. If we are interested in the numerical determination of $P_1(t)$, which is the probability that the particle is in state 1 at time $t \in (t_0, t_{\max})$, we would generate a large number N of those individual trajectories and determine in how many of them the particle is in state 1 at time t. As the particles are independent, we can as well take a global point of view. Instead of running the above programs N times, we run a system consisting of N particles. In order to compute $P_1(t)$, we would count how many particles $n(t)$ there are in state 1 at time t and set $P_1(t) = n(t)/N$.

There are two ways by which we could simulate the simultaneous trajectories of N particles. In the first one, we determine for each one of the N particles the time of the next jump. So, if we start at $t = t_0$ with all particles at state 1 (because we have chosen the initial condition $P_1(t_0) = 1$), then we compute the times of the next jump t_1^k for $k = 1, \ldots, N$. Each of these times is computed as $t_1^k = t_0 - \frac{\log u_0^k}{\omega_{1\to 2}}$, where u_0^k are a set of N independent $\hat{\mathbf{U}}(0, 1)$ uniform random numbers. Next we

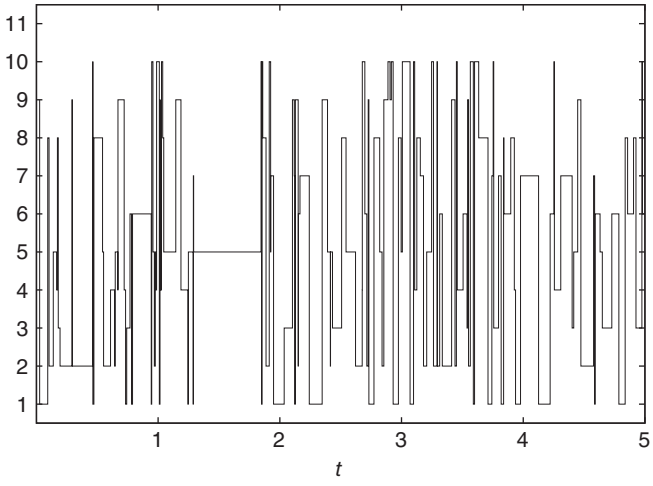


Figure 9.2 Typical trajectory of a stochastic system that jumps between 10 states using the rates $\omega_{i\to j} = |i - j|$.

find the minimum of all these times $t_1 = \min(t_1^1, t_1^2, \ldots, t_1^N)$: say, $t_1^{k_1}$. At this time $t_1 = t_1^{k_1}$, we place the particle k_1 in state 2. We now proceed by computing again the jumping times t_2^k for all $k = 1, \ldots, N$ particles using $t_2^k = t_1 - \frac{\log u_1^k}{\omega_{i\to j}}$, where for each particle $k = 1, \ldots, N$ we have to set the correct rate $\omega_{1\to 2}$ or $\omega_{2\to 1}$ depending on which state this particle is at time t_1. Once we have determined the minimum time $t_2 = \min(t_2^1, \ldots, t_2^N)$, say $t_2 = t_2^{k_2}$, we let the particle k_2 jump from the state it is in to the other. The process continues until the time t_j reaches the desired maximum time.

We now describe the second, more convenient, way of generating trajectories for the N-particle system. As all we need for the calculation of $P_1(t)$ is the number of particles $n(t)$ in state 1 at time t, we adopt the ''occupation numbers'' point of view and characterize the ensemble not by the state every particle is in but directly by the number n of particles in state 1 (and $N - n$ particles in state 2). From this alternative point of view, the variable n can take any of the $N + 1$ values $n = 0, 1, \ldots, N$. So, we consider that the whole ensemble can be in any of $N + 1$ states labeled by the value of n. The situation is formally similar to the M-state case explained before (program `ratem`), but now the $M = N + 1$ possible states are collective states of the whole system of N particles, instead of an individual state. Furthermore, the problem is simpler than the general M-state case explained before, as the only possible transitions allowed are those that increase (respectively, decrease) in one unit the value of n, corresponding to transitions from one particle from 2 to 1 (respectively, from 1 to 2). As found from Equations (8.27) and (8.28) in Section 8.1.2, the rate of the transition from n to $n + 1$ is $\Omega(n \to n + 1) = (N - n)\omega_{2\to 1}$ and the rate of the transition from n to $n - 1$ is $\Omega(n \to n - 1) = n\omega_{1\to 2}$. Then, if at time t_0 we are in a global state characterized by n particles in state 1, we have to compute only the time

$t_{n\to n+1} = -\frac{\log u_0^1}{\Omega(n\to n+1)}$ of the next jump $n \to n+1$ and the time $t_{n\to n-1} = -\frac{\log u_0^2}{\Omega(n\to n-1)}$ of the next jump to $n \to n-1$, using two independent $\hat{\mathbf{U}}(0,1)$ uniform random numbers u_0^1, u_0^2, and implement the action implied by the minimum of these two times, $t_1 = t_0 + \min(t_{n\to n+1}, t_{n\to n-1})$, setting $n \to n+1$ or $n \to n-1$ accordingly. The process then repeats itself, starting at time t_1 finding the time t_2 and the state of the next jump. We now give an example of a program listing that implements this numerical method. In this listing, we have used that, if $n = 0$ (all particles are in state 2), then the only possible transition is to $n = 1$, and that if $n = N$ (all particles in state 1) the only possible transition is to $n = N - 1$ (the number N is indicated in the program by the variable `N0` as capital and lower-case letters can be mistaken by the compiler). We have also chosen the initial state with a random number of particles in state 1: the function `i_ran(N0+1)-1` returns a number between `0` and `N0`.

```
program rate2b
  implicit double precision(a-h,o-z)
  N0=1000
  tmax=100.0d0
  t=0.0d0
  w12=0.5d0
  w21=1.0d0
  n=i_ran(N0+1)-1
  write(66,*) t,n
  do while (t < tmax)
    if (n == 0) then
      tn=-dlog(ran_u())/(N0*w21)
      in=1
    elseif (n ==  N0) then
      tn=-dlog(ran_u())/(N0*w12)
      in=N0-1
    else
      tn1=-dlog(ran_u())/((N0-n)*w21)
      tn2=-dlog(ran_u())/(n*w12)
      if (tn1 < tn2) then
        tn=tn1
        in=n+1
      else
        tn=tn2
        in=n-1
      endif
    endif
    t=t+tn
    write(66,*) t,n
    n=in
    write(66,*) t,n
  enddo
end program rate2b
```

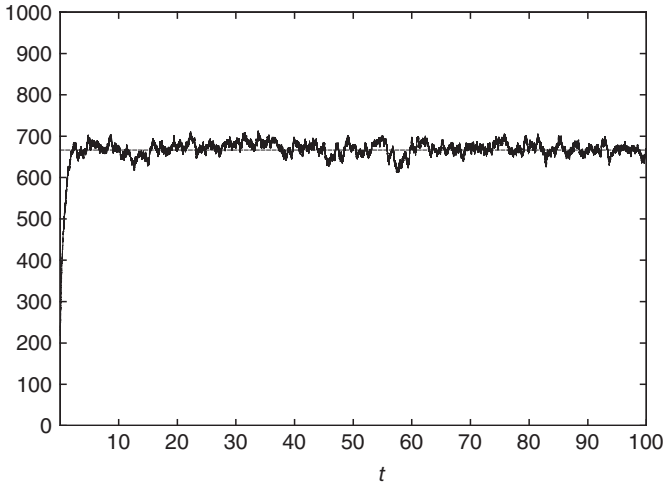


Figure 9.3 Number of particles in state 1 for an ensemble of $N = 1000$ particles that can jump between states 1 and 2 with rates $\omega_{1\to 2} = 0.5$, $\omega_{2\to 1} = 1.0$. The horizontal dashed line is the steady-state value $\frac{n_{\rm st}}{N} = \frac{\omega_{2\to 1}}{\omega_{1\to 2}+\omega_{2\to 1}}$, as found in (8.11).

In Figure 9.3, we plot one output of this program. There is an obvious "practical" advantage of the simulation of the N particles running simultaneously, namely that it is much easier to perform the averages over the N particles. For example, it is obvious from this figure that a steady state is reached in which $P_1(t)$ (given by the ratio of $n(t)$ to the number of particles, $N = 1000$) fluctuates around value equal to the theoretical prediction $\frac{\omega_{2\to 1}}{\omega_{1\to 2}+\omega_{2\to 1}}$.

When the N particles are not independent, that is, when the individual transition rates $\omega_{i\to j}$ depend on the occupation of the states, the simultaneous running of the N particles has another big advantage, namely that the number of particles in each state is easily accessible. Consider, for example, the chemical reaction discussed in Section 8.3. In that example, the individual rate for a Na atom to react (to go from state 1, unbound or free, to state 2, bounded to the molecule) depends on the number n of Cl atoms (we take the simplifying assumption that the number n of Na atoms is equal to the number n of Cl atoms at all times): $\omega_{1\to 2}[n] = k_{12}V^{-1}n$. It would be very costly to program this by finding the times at which individual atoms combine to form a molecule and keeping track of how many atoms n there are bound at a given time t. Instead, we take the occupation numbers point of view and use the global rates given in (8.56) and (8.57): $\Omega(n \to n+1) = (N-n)\omega_{21}$ and $\Omega(n \to n-1) = k_{12}V^{-1}n^2$. In the previous program listing, all we need is to modify the relevant lines to

```
if (n ==  0) then
  tn=-dlog(ran_u())/(N0*w21)
  in=1
elseif (n ==  N0) then
  tn=-dlog(ran_u())/(N0**2*k12/V)
```

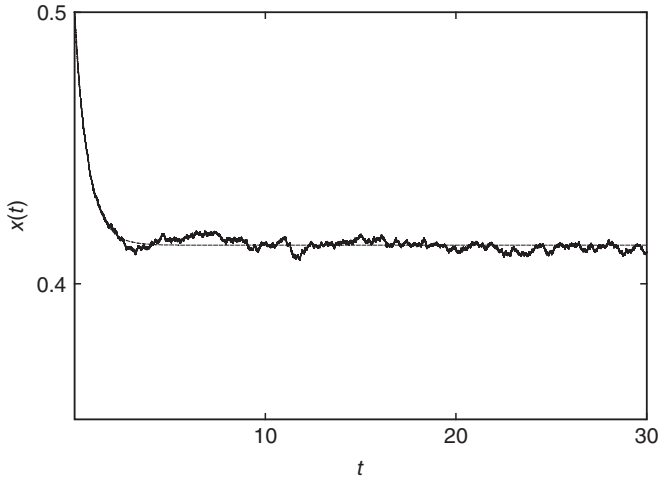


Figure 9.4 Fraction $x(t) = n(t)/V$ of unbound A atoms in the chemical reaction $A + B \rightleftarrows AB$. Parameters: $N = 10^4$, $k_{12} = 0.5$, $V/N \equiv \nu = 2$, $\omega_{21} = 1$. The initial condition is $n_0/V = 0.5$. The analytical line is the result of the mean-field theory, see Exercise 8.6.

```
    in=N0-1
  else
    tn1=-dlog(ran_u())/((N0-n)*w21)
    tn2=-dlog(ran_u())/(n**2*k12/V)
  ..............
```

and define `k12` and `V` somewhere at the beginning of the program. In Figure 9.4, we plot the result of this program and compare it with the predictions of the mean-field theory (see Exercise 8.6).

A simple extension of this algorithm can be used in the case where an individual particle can be in $M > 2$ states. All we need to do is to specify the occupation numbers for particles in each state $(n_1, \ldots, n_M)$ and the possible rates $(n_1, \ldots, n_M) \to (n'_1, \ldots, n'_M)$. As the individual rate to go from state i to state j is $\omega_{i\to j}$, the rate at which the global transition $(n_1, \ldots, n_i, \ldots, n_j, \ldots, n_M) \to (n_1, \ldots, n_i - 1, \ldots, n_j + 1, \ldots, n_M)$ occurs is $n_i\omega_{i\to j}$ because any of the n_i particles in state i can make the jump to state j. The total number of possible transition rates $\Omega((n_1, \ldots, n_M) \to (n'_1, \ldots, n'_M))$ for the global system is $M(M-1)$, which can be a very large number. In practice, however, not all transitions are permitted by the rules of the process. Instead of giving now a specific example, we will explain first a modification that leads to a much more efficient programming of the numerical simulations.

9.2 The Residence Time Algorithm

There is a very simple but very effective modification of the numerical algorithm to simulate a stochastic process and its associated master equation. It bears different

names (residence time, kinetic Monte Carlo, n-fold way or Bortz–Kalos–Lebowitz, dynamic Monte Carlo, or Gillespie algorithm, etc.), as it has been derived independently a number of times with minor variations and emphasizing different applications in each case.

We first consider only one particle. It can be in any of M states and we need to determine the state to which it will jump next and the time of that jump. Assume that, as before, the particle is in the state i_0 at time t_0. First, we compute the rate of escape from this state i_0 to **any other state** $j \neq i_0$. This is nothing but $W_{i_0} = \sum_{j \neq i_0} \omega_{i_0 \to j}$. Then we compute the time interval to the next jump $t_{i_0 \to i_1}$ using this total rate:

$$t_{i_0 \to i_1} = \frac{-\ln u_0}{W_{i_0}}. \tag{9.8}$$

Once the time of the next jump has been determined as $t_1 = t_0 + t_{i_0 \to i_1}$, then we have to determine where to jump, or which is the final state i_1. Recall that $\omega_{i \to j} dt$ is the probability of jumping from i to j in the time interval $(t, t + dt)$, whereas $W_i dt$ is probability of jumping to any state during that same time interval. Therefore, the probability $p_{i_0 \to j}$ of reaching state $j \neq i_0$, knowing that there has been a jump, is the conditional probability

$$p_{i_0 \to j} = \frac{\omega_{i_0 \to j}}{W_{i_0}}. \tag{9.9}$$

In order to determine the final state i_1 according to these probabilities, we use the general technique explained in Section 2.4 to generate the discrete distribution (1.18): draw a random number v_0 uniformly distributed in the interval $(0, 1)$, and find the smallest i_1 that satisfies $\sum_{j=1}^{i_1} p_{i_0 \to j} > v_0$, or equivalently, $\sum_{j=1}^{i_1} \omega_{i_0 \to j} > v_0 W_{i_0}$.

All these ideas can be implemented using the following program, where we use, again, as a specific example, the rates $\omega_{i \to j} = |i - j|$.

```
program ratemg
  implicit double precision(a-h,o-z)
  parameter (M=10)
  dimension w(M,M),wt(M)
  do i=1,M
    wt(i)=0.0d0
    do j=1,M
      w(i,j)=abs(i-j)
      wt(i)=wt(i)+w(i,j)
    enddo
  enddo
  tmax=10.0d0
  t=0.0d0
  i=i_ran(M)
  write(66,*) t,i
```

```
  do while (t < tmax)
    tn=-dlog(ran_u())/wt(i)
    p=0.0d0
    j=0
```

```
    r=ran_u()*wt(i)
    do while (r > p)
    j=j+1
    p=p+w(i,j)
    enddo
    t=t+tn
    write(66,*) t,i
    i=j
    write(66,*) t,i
  enddo
end program ratemg
```

We store in vector `wt(i)` the total rate W_i to escape from state i. The results of this program look, as they should, similar to those displayed in Figure 9.2. Again, in order to obtain meaningful statistics to compute, for example, the probability $P_1(t)$ of being in state 1, we should run that program a large number N of times and average the results. Alternatively, we could use the residence time algorithm to run the N independent particles simultaneously, as we did before.

The second option is to consider the occupation numbers. This method has the advantage that it can also be used if particles are interacting such that the rates depend on the state of the other particles. We take now that there are N (possibly interacting) particles and consider the full set of occupation number variables $(n_1, \dots, n_M)$ that give the number of particles n_k which are on each of the possible states $k = 1, \dots, M$. These variables will change (typically by a small amount), and the rates of the transitions $(n_1, \dots, n_M) \to (n'_1, \dots, n'_M)$ will depend on the variables $(n_1, \dots, n_M)$ themselves, although, typically, not many of these transitions will be allowed. We will give details of the method by using the example of the Lotka–Volterra prey–predator model introduced in Section 8.3.

In the Lotka–Volterra model, the required variables are the number n_1 of live preys and the number n_2 of live predators. In the space (n_1, n_2), and according to the rules of the model, there are three possible transitions whose rates are given in (8.74)–(8.76). For a given state (n_1, n_2), we compute the total escape rate as

$$W(n_1, n_2) = \Omega\left((n_1, n_2) \to (n_1 + 1, n_2)\right) + \Omega\left((n_1, n_2) \to (n_1, n_2 - 1)\right) + \Omega\left((n_1, n_2) \to (n_1 - 1, n_2 + 1)\right). \tag{9.10}$$

As discussed earlier, the time to the next transition will be $-\log(u)/W(n_1, n_2)$, with u being a $\hat{\mathbf{U}}(0, 1)$ random number. Once this transition time has been found, the next step is to decide which one of the three possible transitions will happen. Each one has a probability

$$p_1 = \Omega\left((n_1, n_2) \to (n_1 + 1, n_2)\right)/W, \tag{9.11}$$

$$p_2 = \Omega\left((n_1, n_2) \to (n_1, n_2 - 1)\right)/W, \tag{9.12}$$

$$p_3 = \Omega\left((n_1, n_2) \to (n_1 - 1, n_2 + 1)\right)/W \tag{9.13}$$

and we chose transitions 1, 2, or 3 according to these probabilities. Here is a full program that implements this algorithm for the Lotka–Volterra model.

```
program LotkaVolterra
  implicit double precision(a-h,o-z)
  double precision k1
  tmax=100.0d0
  t=0.0d0
  V=1000
  w0=0.5d0
  k1=1.0d0
  a=k1/V
  w2=0.5d0
  dt=0.1d0
  n1=0.5d0*V
  n2=0.25d0*V
  write(66,*) t,dble(n1)/V,dble(n2)/V
  tw=0.0d0
  do while (t < tmax)
    omega1=w0*n1
    omega2=w2*n2
    omega3=a*n1*n2
    omega=omega1+omega2+omega3
    tn=-dlog(ran_u())/omega
    t=t+tn
    r=ran_u()*omega
    if (r < omega1) then
      n1=n1+1
    else if (r < omega1+omega2) then
      n2=n2-1
    else
      n1=n1-1
      n2=n2+1
    endif
    if (t-tw > dt) then
      write(66,*) t,dble(n1)/V,dble(n2)/V
      tw=t
    endif
  enddo
end program LotkaVolterra
```

In this particular code, the initial condition is set to $n_1(0) = 0.5V$, $n_2(0) = 0.25V$. Note the presence of the variable `dt` which controls the minimum time between the writing of the values of the variables in unit `66`. The only reason to include this variable is that the time between transitions `tn` is very small, mainly for large values of V, and hence we end up with a lot of data that has too much detail. It is enough for most applications to have data spaced over a larger time `dt`. The results of this program can be seen in Figure 9.5 for increasing values of the volume V. Note how both variables giving the concentration of prey $x_1(t) = n_1(t)/V$ and predator $x_2(t) = n_2(t)/V$ oscillate, but there is a delay between them. This is because, when the density of prey is very large, the predators find a lot of food and can increase their offspring by eating the prey. This decreases the density of prey and induces that some time later the predator will find less food and will start decreasing in

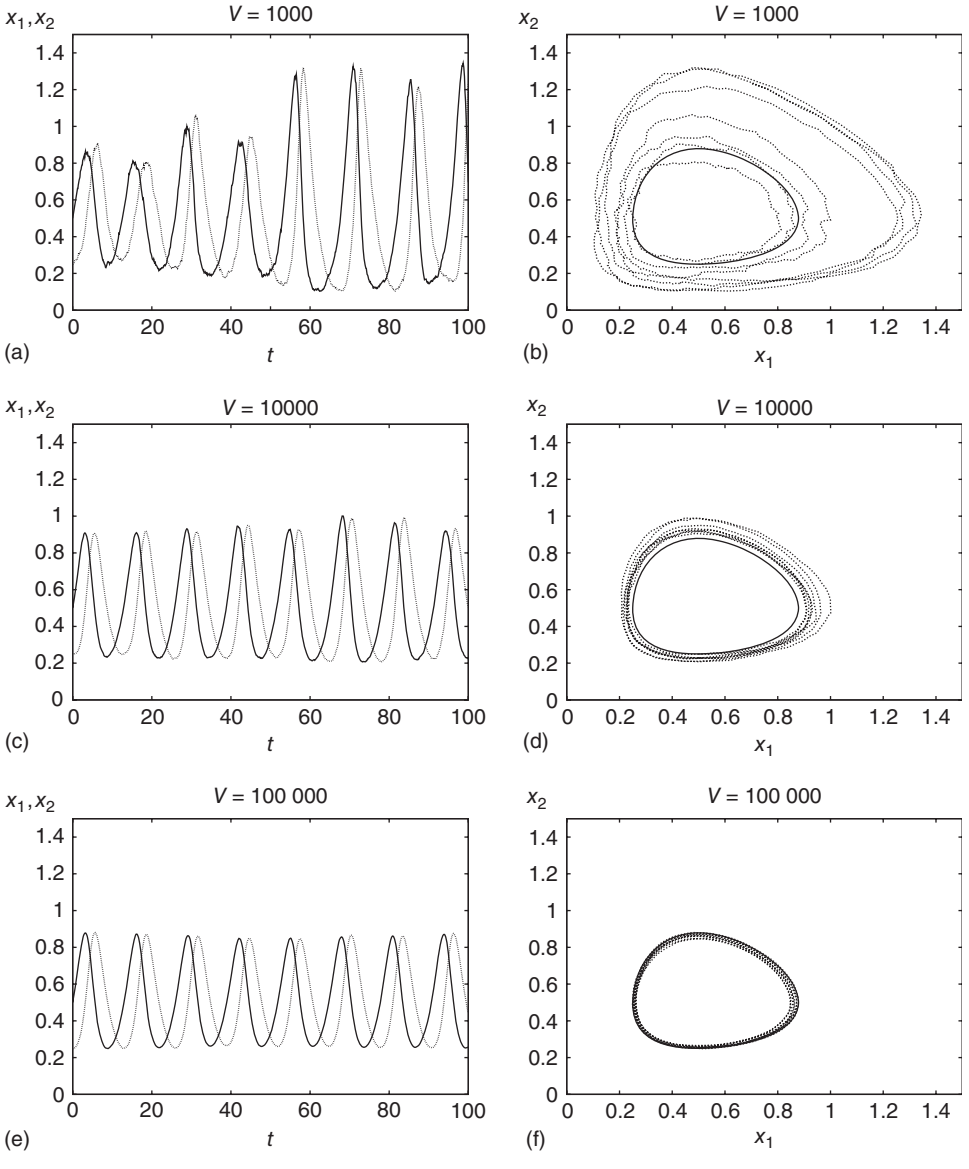


Figure 9.5 Trajectories obtained from the application of the residence time algorithm to the Lotka–Volterra model. In panels (a), (c), and (e), we plot the prey variable $x_1(t)$ (dashed line) and the predator variable $x_2(t)$ (solid line). In panels (b), (d), and (f), we plot the trajectories in the (x_1, x_2) phase space, indicating by a solid thick line the result of the mean-field theory. In (a) and (b), $V = 10^3$; (c) and (d) $V = 10^4$; (e) and (f) $V = 10^5$. Parameters: $\omega_0 = 0.5$, $\omega_1 = 0.5$, $k_1 = 1$, initial condition $x_0 = 0.5$, $y_0 = 0.25$.

number. Then, as there are fewer predators, the prey can increase again. These oscillations in the numbers of prey and predators are well known in ecological populations.

The results of the simulation are compared against the result of the mean-field theory, Equations (8.107) and (8.108). These equations cannot be solved explicitly in terms of simple functions to give $x_1(t)$ and $x_2(t)$, but it is not so difficult to prove that there is a constant of motion (see Exercise 8.7) that allows one to obtain the closed curve in the (x_1, x_2) plane, as depicted in the right panels of Figure 9.5. For a large volume V, the mean-field solution represents the simulation results well, but as V decreases the trajectories, while still oscillatory, depart more and more from the mean-field solution.

Further Reading and References

The name ''first reaction method'' appears in [42]. The first versions of the residence time algorithm (not necessarily with that name) can be found in [43–45].

Exercises

1) Implement the first reaction method and the residence time algorithm to study the birth (from a reservoir) and death process of Section 8.3. Start with $n = 0$ particles and compute the time evolution of the average value $\langle n(t) \rangle$ and its variance $\sigma^2[n(t)]$. Check with the theoretical results obtained by using the generating function. Compare the efficiency of both methods as a function of the rates Ω_A and ω.
2) Implement the residence time algorithm to compute the evolution of the average number of molecules in the chemical reaction $A + B \rightleftarrows AB$. Compare it with the result of the mean-field theory.
3) As a follow-up to Problem 8.8, use the residence time algorithm to compute the average and the variance of the number of intermediate molecular concentration and study under which circumstances (values of the rates) it can be considered a constant as assumed in the Michaelis–Menten theory.
4) Implement the residence time algorithm to study the Lotka–Volterra prey–predator model. Check that the fluctuation in the trajectory scale is $V^{-1/2}$, V being the total area accessible to the population.
5) Use the results of Exercise 8.2 to derive an algorithm that generates the times of the next reaction in the case of a time-dependent rate of the form $\omega(t) = \omega_0 \sin^2(t)$.
6) Consider the SIR model for the spread of epidemics adding a spontaneous contagion rate $S \xrightarrow{\alpha} I$ with a time-dependent rate $\alpha(t) = \alpha_0 \sin^2(\omega t)$, modeling the seasonal variations of the presence of the virus causing the illness. Use the first reaction method and the results of the previous exercise and write

a program to simulate this process. Study the variations of the number of infected people with time.

7) Consider a system composed by N particles. Each particle can be in one of two states and can make random jumps from one state to the other with a rate which is linear with the number of particles in the other state. Therefore, if there are n particles in state 1, the individual rate to jump from 1 to 2 is $a_1 + h_1(N - n)$ and the rate to jump from 2 to 1 is $a_2 + h_2 n$. Write down the evolution equations for the first two moments $\langle n(t)\rangle$ and $\rangle n^2(t)\rangle$ and solve them (making the mean-field assumption if necessary). Use the residence time algorithm and compare the results for the moments and the probabilities $p_{\rm st}(n)$ in the steady state using $a_1 = a_2 = 1$ and $h_1 = h_2 = h$, analyzing separately the cases $h = 1$, $h < 1$, and $h > 1$. This is Kirman's model for the herding behavior in financial markets [46], inspired by the behavior of ants that can chose between two paths when searching for food.

10
Hybrid Monte Carlo

The so-called hybrid Monte Carlo (HMC) algorithm offers a very interesting way to implement the basic proposal–acceptance scheme. The name ''hybrid'' refers to the fact that the proposal is done using techniques borrowed from molecular dynamics, while the acceptance is the usual Metropolis algorithm. This is an example of a *collective update* algorithm, where the proposal configuration implies a change in every one of the system variables while still keeping a reasonable value for the acceptance probability.

Let us first review very briefly the basic concepts we need about molecular dynamics.

10.1
Molecular Dynamics

Molecular dynamics is a very direct approach to the simulation of a system with many degrees of freedom. One deals directly with the microscopic variables $X = (x_1, x_2, \ldots, x_N)$ and their conjugate momenta $P \equiv (p_1, p_2, \ldots, p_N)$. The Hamiltonian $\mathcal{H}(X, P)$ determines the evolution of the dynamical variables after solving the Hamilton's equations

$$\frac{dx_i}{dt} = \frac{\partial \mathcal{H}}{\partial p_i}, \tag{10.1}$$

$$\frac{dp_i}{dt} = -\frac{\partial \mathcal{H}}{\partial x_i}, \quad i = 1, \ldots, N. \tag{10.2}$$

In some cases of interest, it is possible to split the Hamiltonian into potential $V(X)$ and kinetic $\mathcal{T}(P)$ terms with

$$\mathcal{H}(X, P) = V(X) + \mathcal{T}(P) = V(X) + \frac{P^2}{2} = V(X) + \sum_{i=1}^{N} \frac{p_i^2}{2} \tag{10.3}$$

(we take the mass of the particles $m_i = 1$) in which case, Hamilton's equations reduce to

$$\frac{dx_i}{dt} = p_i, \tag{10.4}$$

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

$$\frac{dp_i}{dt} = -\frac{\partial V}{\partial x_i} \equiv F_i \tag{10.5}$$

where $F_i(X)$ is the ith component of the force vector.

In a nutshell, molecular dynamics solves numerically this system of $2N$ coupled differential equations in order to generate trajectories in phase space: $(X(t), P(t)) = (x_1(t), \dots, x_N(t), p_1(t), \dots, p_N(t))$. As Hamilton's equations conserve the Hamiltonian, that is, $\mathcal{H}(X, P) = E$, a constant depending only on initial conditions $(X(0), P(0))$, the generation of these trajectories allows one to sample the microcanonical ensemble. One can then use these trajectories to compute the time averages of any dynamical function $G(X, P)$ as

$$\overline{G} = \lim_{t\to\infty} \frac{1}{t - t_0} \int_{t_0}^{t} dt'\, G\left(X(t'), P(t')\right). \tag{10.6}$$

This temporal average, under appropriate conditions, coincides with the average in the canonical distribution $\overline{G} = \langle G \rangle$, which we have considered in previous chapters. Of course, this is the ideal scenario. In practice, the use of molecular dynamics to sample equilibrium averages has many problems: in addition to the systematic errors coming from the approximate numerical scheme needed to integrate Hamilton's equations, there are statistical errors because one does not integrate the equations for an infinite time as required by (10.6). A somewhat annoying problem with an equilibrium calculation using the microcanonical ensemble is that the temperature T is defined *a posteriori* using, for example, equipartition of the energy, that is

$$\frac{N}{2} kT = \overline{\sum_{i=1}^{N} \frac{p_i^2}{2}}. \tag{10.7}$$

Therefore, the value of the temperature is also subject to statistical and systematic errors and cannot be set exactly from the beginning. It takes some trial and error to set the initial condition $(X(0), P(0))$ such that the resulting temperature coincides with the one we are interested in. Despite these and other problems, molecular dynamics has been very successful and one can count by the thousands the applications of this technique. Moreover, it is particularly useful in the calculation of non-equilibrium dynamical properties that require integration for a finite time.

There exist many algorithms for the numerical integration. They use different formulas to obtain $(X(t + \delta t), P(t + \delta t))$ from $(X(t), P(t))$. Algorithms become more precise as the time step δt decreases, but they require then more computer resources to reach a target time.

The simplest (and arguably the worst) method is Euler algorithm that we learnt in high school (and it seems that this is the only algorithm some people ever learnt). It uses the recurrence relation

$$x_i(t + \delta t) = x_i(t) + \delta t\, p_i(t) + O(\delta t)^2, \tag{10.8}$$

$$p_i(t + \delta t) = p_i(t) + \delta t\, F_i(t) + O(\delta t)^2. \tag{10.9}$$

As indicated, the error in one time step is of order $(\delta t)^2$. After n integration steps, we reach a time $\Delta t = n\delta t$, and the accumulated error scales as $nO(\delta t)^2 = \Delta t O(\delta t)$.

There are more sophisticated numerical methods. A particularly interesting one is the ''leap-frog'' (or Verlet) algorithm. It uses the recurrence relation

$$x_i(t+\delta t) = x_i(t) + \delta t\left(p_i(t) + \frac{\delta t}{2}F_i(t)\right) + O(\delta t)^3, \tag{10.10}$$

$$p_i(t+\delta t) = p_i(t) + \frac{\delta t}{2}\left(F_i(t) + F_i(t+\delta t)\right) + O(\delta t)^3. \tag{10.11}$$

In the second line, we use for the calculation of the force $F(t+\delta t)$ the updated values of the variables $X(t+\delta t)$ that come from the first line of the algorithm. What makes this (and other similar) algorithms interesting is not so much the fact that the error is smaller than in the Euler algorithm, but the fact that leap-frog respects exactly[1)] two characteristic properties of the Hamiltonian dynamics: *reversibility* and *area preservation*.

The reversibility property states that, if starting from the state $(X(t), P(t))$ we reach after a time Δt the state $(X(t+\Delta t), P(t+\Delta t))$ and then we reverse all momenta and consider the evolution from the state $(X(t+\Delta t), -P(t+\Delta t))$, then after a further time Δt we reach back the initial state with the reversed momenta: $(X(t+2\Delta t), P(t+2\Delta t)) = (X(t), -P(t))$. We now prove this property. For the sake of simplicity, we consider the case $N = 1$, but a general proof holds in the more general case of an arbitrary number of degrees of freedom. It is enough to prove this property for a single time step $n = 1$ or $\Delta t = \delta t$, as the case of $n > 1$ follows by a straightforward repetition of this basic proof. So, consider that we start at time t with the initial condition $(x(t), p(t))$ and apply the leap-frog algorithm once to reach time $t+\delta t$ with coordinate and momentum $(x(t+\delta t), p(t+\delta t))$, given by

$$x(t+\delta t) = x(t) + \delta t\left(p(t) + \frac{\delta t}{2}F(t)\right), \tag{10.12}$$

$$p(t+\delta t) = p(t) + \frac{\delta t}{2}\left(F(t) + F(t+\delta t)\right). \tag{10.13}$$

Then, we reverse the momentum and start with an initial condition $(x(t+\delta t), -p(t+\delta t))$. Starting from this value, we apply the leap-frog algorithm to reach time $t+2\delta t$ as

$$x(t+2\delta t) = x(t+\delta t) + \delta t\left(-p(t+\delta t) + \frac{\delta t}{2}F(t+\delta t)\right), \tag{10.14}$$

$$p(t+2\delta t) = -p(t+\delta t) + \frac{\delta t}{2}\left(F(t+\delta t) + F(t+2\delta t)\right). \tag{10.15}$$

From (10.13), we derive $-p(t+\delta t) + \frac{\delta t}{2}F(t+\delta t) = -p(t) - \frac{\delta t}{2}F(t)$, which when replaced in (10.14) leads to $x(t+2\delta t) = x(t+\delta t) + \delta t(-p(t) - \frac{\delta t}{2}F(t))$, and again by use of (10.12) reduces immediately to $x(t+2\delta t) = x(t)$. This, in turn, implies $F(t+2\delta t) = F(t)$, as all dependence on t of the force F is through the coordinate variable $F(t) = F(x(t))$. This reduces (10.15) to $p(t+2\delta t) = -p(t+\delta t) + \frac{\delta t}{2}(F(t+\delta t) + F(t))$ which, after (10.13), reduces to $p(t+2\delta t) = -p(t)$, completing the proof of time reversibility for the leap-frog algorithm.

1) Except unavoidable numerical rounding-off errors.

The area-preserving or, in other words, Liouville theorem tells us that, if we consider the evolution $(X(t), P(t)) \to (X(t+\Delta t), P(t+\Delta t))$ as a change of variables, then the determinant of the Jacobian matrix of this change is equal to 1, or

$$\left| J\left(\frac{X(t+\Delta t), P(t+\Delta t)}{X(t), P(t)} \right) \right| = 1. \tag{10.16}$$

Again, it is sufficient to prove this identity for a single recurrence of the leap-frog algorithm, $n = 1$, as the case $n > 1$ follows from the fact that the determinant of the Jacobian after n steps is the product

$$\left| J\left(\frac{X(t+\Delta t), P(t+\Delta t)}{X(t), P(t)} \right) \right| = \prod_{k=0}^{n-1} \left| J\left(\frac{X(t+(k+1)\delta t), P(t+(k+1)\delta t)}{X(t+k\delta t), P(t+k\delta t)} \right) \right|. \tag{10.17}$$

Again, for simplicity, we consider the case of a single degree of freedom, $N = 1$. The Jacobian of the transformation (10.12) and (10.13) is explicitly given as

$$J = \begin{pmatrix} \frac{\partial x(t+\delta t)}{\partial x(t)} & \frac{\partial x(t+\delta t)}{\partial p(t)} \\ \frac{\partial p(t+\delta t)}{\partial x(t)} & \frac{\partial p(t+\delta t)}{\partial p(t)} \end{pmatrix}. \tag{10.18}$$

To perform the derivatives, one must not forget that $F(t) = F(x(t))$. Using the chain rule and (10.12), we obtain

$$\frac{\partial x(t+\delta t)}{\partial x(t)} = 1 + \frac{(\delta t)^2}{2} F'(t). \tag{10.19}$$

Similarly,

$$\frac{\partial p(t+\delta t)}{\partial x(t)} = \frac{\delta t}{2} \left(F'(t) + F'(t+\delta t) \frac{\partial x(t+\delta t)}{\partial x(t)} \right) \tag{10.20}$$

$$= \frac{\delta t}{2} \left(F'(t) + F'(t+\delta t) \left(1 + \frac{(\delta t)^2}{2} F'(t) \right) \right). \tag{10.21}$$

The next derivatives are as follows:

$$\frac{\partial x(t+\delta t)}{\partial p(t)} = \delta t, \tag{10.22}$$

$$\frac{\partial p(t+\delta t)}{\partial p(t)} = 1 + \frac{\delta t}{2} F'(t+\delta t) \frac{\partial x(t+\delta t)}{\partial p(t)} \tag{10.23}$$

$$= 1 + \frac{(\delta t)^2}{2} F'(t+\delta t). \tag{10.24}$$

Replacing these results in (10.18), one obtains that $|J| = 1$. An algorithm that conserves this important property of the Hamiltonian dynamics is called ''symplectic.'' Leap-frog is the simplest member of the family of symplectic algorithms.

Let us stress again that, as the recurrence relation given by (10.10) and (10.11) satisfies exactly and for any arbitrary value of δt the properties of reversibility and area-preserving, it satisfies the same properties after n iterations of the basic step to reach a time $\Delta t = n\delta t$.

Alas, the leap-frog algorithm does not respect another important property of Hamiltonian dynamics: conservation of energy. Therefore,

$$\delta\mathcal{H} = \mathcal{H}(t+\delta t) - \mathcal{H}(t) = O(\delta t)^3 \neq 0 \tag{10.25}$$

or, after n updates, $\Delta\mathcal{H} = \mathcal{H}(t+\Delta t) - \mathcal{H}(t) \neq 0$, that is, the initial energy E is not conserved during the time evolution. Some tricks can be used to enforce energy conservation (like multiplying all momenta by a common factor after every one of a number of time steps) but this does not concern us here. This ends our brief presentation of molecular dynamics.

10.2 Hybrid Steps

Let us go back to the problem of sampling the canonical distribution $f(X) = \mathcal{Z}^{-1}e^{-\beta V(X)}$. We denote now by $X = (x_1, \ldots, x_N)$ the full set or variables of interest. These variables can be positions, angles, or even velocities and momenta. The idea of the HMC method is to use a Metropolis scheme with a proposal $g(X \to X')$ that relies on a *fake* Hamiltonian dynamics in which X is the initial condition and X' the result of the dynamical evolution. We stress that the Hamiltonian dynamics is fake; it has nothing to do with the real Hamiltonian dynamics that the system X might satisfy (if any). Similarly, $V(X)$ itself can be a true Hamiltonian function or not.

To define this dynamics, we define the auxiliary variables $P = (p_1, \ldots, p_N)$ for each and every one of the original variables $x = (x_1, \ldots, x_N)$. These, we will call ''momenta'' variables, but bear in mind that they are auxiliary variables without any physical meaning. If, by any chance, x_i is already a momenta-type variable, we still define p_i as its associated ''momentum.''

Next we add a ''kinetic energy'' term to define the (fake) Hamiltonian:

$$\hat{\mathcal{H}}(X,P) = V(X) + \frac{P^2}{2} = V(X) + \sum_{i=1}^{N}\frac{p_i^2}{2}, \tag{10.26}$$

$$\frac{dx_i}{dt} = p_i, \tag{10.27}$$

$$\frac{dp_i}{dt} = F_i = -\frac{\partial V}{\partial x_i}. \tag{10.28}$$

To propose a value X', we integrate numerically these equations for a time $\Delta t = n\delta t$, where n and δt are parameters of the method to be tuned for optimal behavior. For the integration, we take an initial condition $X(0)$ and generate random values of the momenta $P(0)$ from a Gaussian distribution $e^{-\frac{\beta}{2}P^2} = e^{-\frac{\beta}{2}\sum_{i=1}^{N}p_i^2}$. That is, each variable p_i is obtained independently of the others from a Gaussian distribution of mean 0 and variance $1/\beta$. It will be clear in a moment why we choose this initial condition for the momenta variables. After application of the numerical algorithm, we obtain the values $(X(\Delta t), P(\Delta t)) \equiv (X', P')$. The basic idea

of the hybrid method is to consider that $X' = X(\Delta t)$ is the proposal that we will accept with the appropriate probability $h(X'|X)$. Let us work out first the pdf $g(X'|X)$. From its definition, $g(X'|X)dX'$ is the probability of proposing a value of the variables in the interval $(X', X' + dX')$. For that, we need to select the right momentum $P(0)$ (whatever it is) in the right interval $(P, P + dP)$ such that the variables at time $t + \Delta t$ will have the required value. As the momenta variables are extracted from a Gaussian distribution, this has a probability $e^{-\frac{\beta}{2}P^2}dP$, hence

$$g(X'|X)dX' = dPe^{-\frac{\beta}{2}P^2} \tag{10.29}$$

with P being the necessary set of momenta to reach X' from X. Similarly, the inverse proposal

$$g(X|X')dX = dP''e^{-\frac{\beta}{2}P''^2} \tag{10.30}$$

where P'' is the necessary set of momenta to reach X' from X in a time Δt. Now comes one key point: if the numerical integrator is time reversible (as leap-frog is), the necessary momenta for the inverse proposal are $P'' = -P'$. Let us write now the detailed balance condition (4.38).

$$f(X)g(X'|X)h(X'|X) = f(X')g(X|X')h(X|X'). \tag{10.31}$$

Replacing (10.29) and (10.30) and using $f(X) = \mathcal{Z}^{-1}e^{-\beta V(X)}$ and $\hat{\mathcal{H}}(X, P) = V(X) + \frac{P^2}{2}$, we obtain after some rearranging

$$e^{-\beta\hat{\mathcal{H}}(X,P)}h(X'|X)dXdP = e^{-\beta\hat{\mathcal{H}}(X',P')}h(X|X')dX'dP' \tag{10.32}$$

where we have also used $P'' = -P'$, $(P'')^2 = (P')^2$, and the corresponding Jacobian $dP'' = dP'$.

You might have seen by now the second key point: if the numerical integration method satisfies the property of area-preserving (as leap-frog does), then $dXdP = dX'dP'$, and the equation to determine the acceptance probability h has a familiar aspect, that is

$$e^{-\beta\hat{\mathcal{H}}(X,P)}h(X'|X) = e^{-\beta\hat{\mathcal{H}}(X',P')}h(X|X'). \tag{10.33}$$

We know how to solve this functional equation. A possible solution is the choice by Metropolis *et al.*:

$$h(X'|X) = \min(1, e^{-\beta\Delta\hat{\mathcal{H}}}), \qquad \Delta\hat{\mathcal{H}} = \hat{\mathcal{H}}(X', P') - \hat{\mathcal{H}}(X, P). \tag{10.34}$$

Before commenting on the advantages of the just-described HMC algorithm, let us summarize: to sample the distribution $f(X) = \mathcal{Z}^{-1}e^{-\beta V(X)}$, follow the steps:

(0) Set an initial condition $X = X(0)$.
(1) Generate a set of values $P(0) = (p_1, \dots, p_N)$ independently from a Gaussian distribution of zero mean and variance $1/\beta$.
(2) Integrate the equations of motion (10.27) and (10.28) for a time $\Delta t = n\delta t$ using any time reversible, area-preserving algorithm (e.g., leap-frog). Let $X' = X(\Delta t)$ and $P' = P(\Delta t)$. Compute $\Delta\hat{\mathcal{H}} = \hat{\mathcal{H}}(X', P') - \hat{\mathcal{H}}(X, P)$.

(3) Accept X' with the chosen probability. For example, $h = \min(1, e^{-\beta\Delta\hat{\mathcal{H}}})$. If accepted, set $X(0) = X'$; otherwise, set again $X(0) = X$.

Repeat steps (1–3). n and δt are parameters of the method that need to be chosen, as usual, to optimize the performance of the method: that is, to reduce the time needed to achieve a given statistical error.

What are the advantages of using such an elaborated proposal? Compare it with the possible alternative: select one single variable x_i, and propose a change to a value x_i' randomly chosen from the interval $(x_i - \Delta, x_i + \Delta)$. Compute the change $V(X') - V(X)$ and accept this proposal with a probability $h = \min(1, e^{-\beta(V(X')-V(X))})$. In order to keep the average acceptance probability to a reasonable value (not too small), we need to keep the average change of $V(X)$ small, which implies a small value for Δ, implying large correlations between the different configurations. This was, in fact, the major problem with the implementation of the Metropolis algorithm that we discussed in the previous chapters.

HMC, on the other hand, uses in the acceptance step a configuration X' in which each and every variable in X has been changed following its evolution during a time Δt. Even though the change in the configurational space X is large, the average acceptance probability can be kept to an acceptable value, because the change $\Delta\hat{\mathcal{H}} = \hat{\mathcal{H}}(X', P') - \hat{\mathcal{H}}(X, P)$ can be kept small. This is so because the only origin of $\Delta\hat{\mathcal{H}}$ not being exactly equal to zero is the numerical errors of the integration algorithm, and those can be controlled by choosing δt sufficiently small.

Another advantage is that the number of calculations of the function $V(X)$ is greatly reduced. This is important when this function requires a large computation time. This is, for example, the case of some quantum field theories where HMC is routinely used. But there are also many applications to other problems of statistical mechanics.

10.3 Tuning of Parameters

We start from the relation

$$\langle e^{-\beta\Delta\hat{\mathcal{H}}} \rangle_{\text{st}} = 1 \tag{10.35}$$

which holds for the average value for the proposed values of $\Delta\hat{\mathcal{H}}$ at the steady state once the configurations are representative of equilibrium, that is, after the necessary thermalization steps, and are distributed according to the distribution $e^{-\beta\hat{\mathcal{H}}}$. The proof of this relation is left as an exercise and it follows the lines of the equivalent result (5.24). Similarly, and following the same line of reasoning that led to (5.26), it turns out that the average acceptance probability is

$$\langle h(X'|X) \rangle_{\text{st}} = \text{erfc}\left(\frac{\sqrt{\beta\langle\Delta\hat{\mathcal{H}}\rangle}}{2} \right) \tag{10.36}$$

and, therefore, decreases (exponentially) with $\langle \Delta\hat{\mathcal{H}} \rangle$ (we assume all averages to be in the steady state now). The whole idea of the HMC method is to keep $\langle \Delta\hat{\mathcal{H}} \rangle$ small, so the average acceptance probability is not too small. First, we compute $\langle \delta\hat{\mathcal{H}} \rangle$, the value of $\langle \Delta\hat{\mathcal{H}} \rangle$ for $\Delta t = \delta t$ or $n = 1$ steps before applying the acceptance step. For the sake of clarity, we restrict ourselves to the case of a single degree of freedom, $N = 1$, but a similar proof holds in the general case. We begin from

$$\delta\hat{\mathcal{H}} = V\left(x(t+\delta t)\right) + \frac{p\,(t+\delta t)^2}{2} - V\left(x(t)\right) - \frac{p\,(t)^2}{2} \tag{10.37}$$

and we use now the evolution equations of leap-frog (10.10) and (10.11), after replacing $F(t) = -V'(x(t))$:

$$x(t+\delta t) = x(t) + \delta t p(t) - \frac{\delta t^2}{2} V'(x(t)) + O(\delta t)^3, \tag{10.38}$$

$$p(t+\delta t) = p(t) - \frac{\delta t}{2} V'(x(t)) - \frac{\delta t}{2} V'(x(t+\delta t)) + O(\delta t)^3. \tag{10.39}$$

Note that in the second equation we must use $x(t+\delta t)$ as given from the first equation. We replace these two equations in (10.37) and expand in a series Taylor in δt up to order $O(\delta t)^6$. After a tedious calculation, one obtains

$$\begin{aligned}\delta\hat{\mathcal{H}} = {} & \frac{(\delta t)^3}{12}\left[3pV'V'' - p^3V^{(3)}\right] \\ & + \frac{(\delta t)^4}{24}\left[-3(V')^2V'' + 3p^2\left((V'')^2 + 2V'V^{(3)}\right) - p^4V^{(4)}\right] \\ & + \frac{(\delta t)^5}{80}\left[-p\left(10V'(V'')^2 + 15(V')^2V^{(3)}\right) + 10p^3\left(V''V^{(3)} + V'V^{(4)}\right) - p^5V^{(5)}\right] \\ & + O(\delta t)^6,\end{aligned} \tag{10.40}$$

with $p = p(t)$, $V' = V'(x(t))$, and so on. To take the average of this quantity, we first notice that the values of p are chosen from a Gaussian distribution of zero mean and variance $1/\beta$ independently on the values of any other quantities. Hence, after taking the averages, we can replace $\langle p \rangle = \langle p^3 \rangle = \langle p^5 \rangle = 0$, $\langle p^2 \rangle = 1/\beta$, and $\langle p^4 \rangle = 3/\beta^2$:

$$\langle \delta\hat{\mathcal{H}} \rangle = \frac{(\delta t)^4}{24}\left\langle -3(V')^2V'' + \frac{3}{\beta}\left((V'')^2 + 2V'V^{(3)}\right) - \frac{3}{\beta^2}V^{(4)}\right\rangle + O(\delta t)^6. \tag{10.41}$$

The averages of the function $V(x)$ must be performed with respect to the equilibrium distribution $\mathcal{Z}^{-1}e^{-\beta V}$, that is

$$\begin{aligned}\langle \delta\hat{\mathcal{H}} \rangle = {} & \frac{(\delta t)^4}{24\mathcal{Z}}\int dx e^{-\beta V}\left(-3(V')^2V'' + \frac{3}{\beta}\left((V'')^2 + 2V'V^{(3)}\right) - \frac{3}{\beta^2}V^{(4)}\right) \\ & + O(\delta t)^6\end{aligned} \tag{10.42}$$

which, after a straightforward manipulation, becomes

$$\begin{aligned}\langle \delta\hat{\mathcal{H}} \rangle &= \frac{(\delta t)^4}{8\mathcal{Z}}\int dx \frac{d}{dx}\left[e^{-\beta V}\left(\frac{V'V''}{\beta} - \frac{V^{(3)}}{\beta^2}\right)\right] + O(\delta t)^6 \\ &= \frac{(\delta t)^4}{8\mathcal{Z}}\left[e^{-\beta V}\left(\frac{V'V''}{\beta} - \frac{V^{(3)}}{\beta^2}\right)\right]_{x=-\infty}^{x=+\infty} + O(\delta t)^6.\end{aligned} \tag{10.43}$$

The integral vanishes at the extremes of the integration interval if we assume that the potential decays sufficiently fast at $x = \pm\infty$. This yields

$$\langle \delta\hat{\mathcal{H}} \rangle = O(\delta t)^6$$

completing the proof. We see that, although each separate trajectory has, in principle, an integration error of order $(\delta t)^3$, the average error in the Hamiltonian is much smaller, of order $(\delta t)^6$. One still has to take into account that, for a system with N degrees of freedom, the Hamiltonian is extensive, of order N, and this is reflected in the prefactor leading to an estimation $\langle \delta\hat{\mathcal{H}} \rangle = NO(\delta t)^6$. Therefore, if we want to keep the acceptance probability constant, we have to scale the time interval of the integration as $\delta t \sim N^{-1/6}$.

10.4 Relation to Langevin Dynamics

Let us now approach the generation of representative configurations according to the Boltzmann factor $e^{-\beta V(X)}$ from a completely different point of view. Imagine we introduce an *ad hoc* stochastic dynamics given by the Langevin equation

$$\frac{\partial x_i(\tau)}{\partial \tau} = -\frac{\partial V}{\partial x_i} + \sqrt{\frac{2}{\beta}}\xi_i(\tau) = F_i(\tau) + \sqrt{\frac{2}{\beta}}\xi_i(\tau), \tag{10.44}$$

where we denote the time variable by τ. Again, this stochastic dynamics might or might not be related to any physical process satisfied by the system $X(t)$. We just take it as a convenient mathematical tool. As usual, the white noise terms have a Gaussian distribution of zero mean and correlations

$$\langle \xi_i(\tau)\xi_j(\tau') \rangle = \delta_{ij}\delta(\tau - \tau'). \tag{10.45}$$

Proceeding as indicated in Section 6.7 one can write down the equivalent Fokker-Planck equation for the Langevin Equation (10.44). The stationary solution of this equation is the canonical distribution $P_{\rm st}(X) = \mathcal{Z}^{-1}e^{-V(X)}$ (see Exercise 6.9 with S being the identity matrix). Therefore, all we need to do to sample this distribution is to integrate (numerically) the Langevin equation and generate trajectories $X(t)$. The different values along this trajectory, after an initial transient, will be distributed according to $P_{\rm st}(X)$. For the numerical integration, we can use, for example, the simple Euler-Maruyama algorithm (7.37)

$$x_i(\tau + \delta\tau) = x(\tau) + \delta\tau F_i(\tau) + \sqrt{\frac{2\delta\tau}{\beta}}u_i(\tau), \tag{10.46}$$

where $u_i(\tau)$ are independent Gaussian variables of mean 0 and variance 1. We know that this is an approximation (and not an extremely good one) to the numerical integration of the Langevin equation. Therefore, if we use the resulting values of $X(t)$, there would be, in addition to the unavoidable statistical errors, some systematic errors of order $(\delta\tau)^{3/2}$ coming from the performance of the Euler algorithm. But now an interesting comparison shows up. Compare the numerical algorithm as given by (10.46) with the one-step leap-frog (10.12) used in the

hybrid algorithm, for $n = 1$, where we refresh the momenta $p_i(t)$ from a Gaussian distribution of zero mean and variance $2/\beta$ at every step:

$$x_i(t + \delta t) = x_i(t) + \frac{\delta t^2}{2} F_i(t) + \delta t p_i(t). \tag{10.47}$$

It turns out that (10.46) and (10.47) are exactly the same if we identify $\delta\tau = \delta t^2/2$ and recall that $p_i(t)$ is a Gaussian random variable of variance $1/\beta$. Remember that both t and τ are, as far as the objective of sampling the Boltzmann distribution is concerned, auxiliary time variables. So we conclude that the basic evolution step of HMC is the same that that of the numerical integration of the Langevin equation. But there is an important addition: the inclusion of the acceptance/rejection step in the hybrid method eliminates the systematic errors of the Euler integrator of the Langevin equation!

10.5 Generalized Hybrid Monte Carlo

We have seen that the HMC method relies on a time reversible, area-preserving map that has a controlled variation of the Hamiltonian. A natural candidate (as we have seen) is the leap-frog integrator of Hamilton equations. It turns out that there are other possibilities.

Remember that our goal is to generate values of the set of stochastic variables $X = (x_1, \dots, x_N)$ distributed according to the Boltzmann factor $f_{\hat{\mathbf{x}}}(X) = \mathcal{Z}^{-1} e^{-\beta V(X)}$. Let us now introduce for each variable $x_i(t)$ a D-dimensional vector of associated momenta $\vec{p}_i = (p_i^1, \dots, p_i^D)$. We consider the following generalized dynamical equations:

$$\frac{dx_i}{dt} = \sum_{s=1}^{D} \sum_{j=1}^{N} (\mathcal{A}^s)_{ij} p_j^s, \quad i = 1, \dots, N, \tag{10.48}$$

$$\frac{dp_i^s}{dt} = \sum_{j=1}^{N} (\mathcal{A}^s)_{ji} F_j, \quad i = 1, \dots, N,\ s = 1, \dots, D \tag{10.49}$$

where, as before, $F_i = -\partial V/dx_i$, and $(\mathcal{A}^s)_{ij}$ are the elements of a set of D arbitrary $N \times N$ matrices, $\mathcal{A}^1, \dots, \mathcal{A}^D$. We can use a compact matrix notation for this set of dynamical equations

$$\frac{dX}{dt} = \sum_{s=1}^{D} \mathcal{A}^s P^s, \tag{10.50}$$

$$\frac{dP^s}{dt} = (\mathcal{A}^s)^{\intercal} F, \quad s = 1, \dots, D, \tag{10.51}$$

with $P^s = (p_1^s, \dots, p_N^s)$ and $P = (P^1, \dots, P^D)$. First of all, it is clear that these equations contain Hamilton's equations in the particular case $D = 1$ and $\mathcal{A} = \mathcal{I}$, the identity matrix. Furthermore, the equations have the property that they conserve the "energy": that is, the function

$$\hat{\mathcal{H}}(X, P) = V(X) + \sum_{i=1}^{N} \sum_{s=1}^{D} \frac{(p_i^s)^2}{2} \tag{10.52}$$

is a constant during the dynamical evolution given by (10.48) and (10.49):

$$\frac{d\hat{\mathcal{H}}}{dt} = 0. \tag{10.53}$$

The leap-frog integrator

$$X(t+\delta t) = X(t) + \delta t \sum_{s=1}^{D} \mathcal{A}^s P^s(t) + \frac{(\delta t)^2}{2} \sum_{s=1}^{D} \mathcal{A}^s (\mathcal{A}^s)^\intercal F(t), \tag{10.54}$$

$$P^s(t+\delta t) = P^s(t) + \frac{\delta t}{2} (\mathcal{A}^s)^\intercal \left(F(t) + F(t+\delta t)\right) \tag{10.55}$$

satisfies, as before, the properties of area-preserving and time reversibility. Therefore, we can set $P(0)$ as a set of independent Gaussian variables of mean 0 and variance 1 and then use the numerical integration (10.54) and (10.55) n times to realize the map $(X(0), P(0)) \to (X(\Delta t), P(\Delta t))$ with $\Delta t = n\delta t$. The obtained variables $X(t+\Delta t)$ are then used as the proposal X', which is accepted according to, for example, the Metropolis *et al.*'s choice $h(X'|X) = \min(1, e^{-\beta \Delta \hat{\mathcal{H}}})$.

It is also possible to connect this generalized HMC method with the Langevin equation. As shown in Exercise 6.9, for a general class of matrices $\mathcal{A}$, the set of Langevin equations

$$\frac{dx_i(\tau)}{d\tau} = \sum_{j=1}^{N} \left[-(\mathcal{A}\mathcal{A}^\intercal)_{ij} \frac{\partial V}{\partial x_j} + \sqrt{\frac{2}{\beta}} \mathcal{A}_{ij} \xi_j \right] \tag{10.56}$$

also have $e^{-\beta V(X)}$ as stationary solution. Therefore, a numerical integration using, for example, the simple Euler-Maruyama scheme

$$x_i(\tau + \delta\tau) = x_i(\tau) + \sum_j \left[-\delta\tau (\mathcal{A}\mathcal{A}^\intercal)_{ij} \frac{\partial V}{\partial x_j} + \sqrt{\frac{2\delta\tau}{\beta}} A_{ij} u_j \right] \tag{10.57}$$

should lead asymptotically to the generation of configurations X distributed according to the Boltzmann factor $e^{-\beta V(X)}$. However, this sampling suffers from numerical errors as the Euler scheme is not a perfect integrator. As before, we can relate the leap-frog integrator (10.54) and (10.55) for $D = 1$ to this numerical scheme (10.57) if we identify $(\delta t)^2/2 = \delta\tau$. Therefore, the generalized HMC method makes exact (in the sense that averages are not biased by the choice of the time step) the numerical integration of the Langevin equation using a matrix time step. Different choices of the matrices $\mathcal{A}^1, \ldots, \mathcal{A}^D$ have been used in the literature. For instance, one can choose $D = 1$ and a matrix $\mathcal{A}$, which is diagonal in Fourier space, to implement the scheme of Fourier acceleration. The choice $D = d$ (the spatial dimension) and $\mathcal{A}^s$ equal to the sth component of the gradient operator can be used to study systems with a conserved order parameter.

Further Reading and References

There are many books covering the field of molecular dynamics simulations. Some of the most well known are [15, 47–49].

HMC methods were introduced in [50]. The generalized method was developed in [51], with applications to system with a conserved order parameter. Equivalent methods from the point of view of the Langevin equation with respect to Fourier acceleration are treated in [52, 53].

Exercises

1) Prove that the leap-frog algorithm satisfies the time-reversal and area-preserving properties in the general case of $N > 1$ number of degrees of freedom.
2) Prove that for the hybrid Monte Carlo algorithm, $\langle e^{-\beta \Delta \hat{\mathcal{H}}} \rangle = 1$, with $\Delta \hat{\mathcal{H}}$ being the proposed change of the (fake) Hamiltonian. Hint: follow the proof given at the end of Chapter 5, but without the symmetry assumption $g(X|Y) = g(Y|X)$.
3) Prove (10.40). You might consider using a symbolic manipulation program such as Mathematica, where the relevant command would be

$$\texttt{Series}\Big[\texttt{V}\Big[\texttt{x} + \texttt{h}\,\texttt{p} - \frac{\texttt{h}^2}{2}\texttt{V}'[\texttt{x}]\Big] - \texttt{V}[\texttt{x}] + \frac{1}{2}\Big(\Big(\texttt{p} - \frac{\texttt{h}}{2}(\texttt{V}'[\texttt{x}] + \texttt{V}'[\texttt{x} + \texttt{h}\,\texttt{p} - \frac{\texttt{h}^2}{2}\texttt{V}'[\texttt{x}]\Big)\Big)^2 - \texttt{p}^2\Big), \{\texttt{h}, 0, 5\}\Big].$$

4) Use the hybrid Monte Carlo algorithm to sample the distribution $f_{\hat{\mathbf{x}}}(x) = Ce^{-\frac{1}{2}x^2 - \frac{1}{4}x^4}$. Determine the optimal value of the time step δt and the number of steps n before the acceptance procedure. Compare the efficiency of this method with the rejection algorithm developed in Section 3.6.
5) Write the Gamma distribution (1.59) with the scale parameter $\theta = 1$ in the form $f_{\hat{\mathbf{x}}}(x) = Ce^{-V(x)}$ and use the hybrid Monte Carlo algorithm to sample this distribution. Find the optimal values of the time step δt and the number of steps n before the acceptance procedure as a function of the parameter α.
6) Apply the hybrid Monte Carlo algorithm to the lattice Φ^4 model described in Section 5.6. Determine the optimal value of the parameters n, δ in order to reduce the correlation time in the critical region.
7) Prove that the dynamical equations (10.50) and (10.51) satisfy exactly the property of time reversibility and conservation of energy. Prove that the same properties are respected (up to round-off numerical errors) by the leap-frog integrator (10.54) and (10.55).
8) Implement the generalized hybrid Monte Carlo method using $D = 1$ and a matrix $\mathcal{A}$ of coefficients $\mathcal{A}_{ij} = e^{\frac{2\pi i}{N} ij}$, with N being the number of degrees of freedom for the pdf given by the free Lagrangian $\mathcal{L}_0$ of (B.4). Evaluate numerically the correlation time τ and discuss its dependence on N.

11
Stochastic Partial Differential Equations

Partial differential equations (PDEs) appear in many fields of science when describing the dynamics in spatially extended systems. There are many kinds of PDEs and several ways to classify them. From a broad perspective, one can distinguish between initial value problems and boundary value problems. The former are associated with the description of the temporal evolution of a spatially extended system starting from a given initial condition, while the latter are associated with the stationary behavior of a variable subject to given boundary conditions. Since a PDE is not fully defined until the boundary conditions are given, boundaries do play a role in both problems. The difference is that, in initial value problems, time plays a relevant role, while that is not the case for boundary value problems. In this chapter, we will focus on initial value problems.

Let us start with some considerations on PDEs to describe initial value problems in deterministic, spatially extended systems. PDEs describe the continuous evolution in time of a continuous field in space. Since computers are digital, it is necessary to discretize both time and space. There are several ways to do so. The most intuitive one is finite differences in which space is discretized in a regular lattice. From a computational point of view, one considers that the field is fully described by the values it takes at the lattice points. Since the field is considered to be smooth, the value of the field at the lattice point is representative of the value of the field over all the discretization interval. Space derivatives are replaced by finite differences, so that the PDE is converted to a set of coupled ordinary differential equations. Then, one can, in principle, use any standard method for ordinary differential equations to integrate it numerically.

Now, consider that the system is under the influence of noise and that the noise at different spatial points is uncorrelated. The noise is not a smooth field in space and time, and therefore the value it takes at a particular point is not representative of the value of the field over the discretization interval. In fact, the situation is even worse for white noise, where the value that the field takes at a precise point at a given time is not even defined since it is a Gaussian number with infinite variance. Therefore, one needs to proceed in a different way in order to obtain a discrete representation of the field and the noise which is consistent and makes sense.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

11.1 Stochastic Partial Differential Equations

Consider a field whose dynamics is given by a PDE and which is subject to the influence of a stochastic field $\xi(\vec{x}, t)$. More precisely, we will consider that the dynamics is given by a PDE of the form:

$$\frac{\partial A(\vec{x}, t)}{\partial t} = q(A, \vec{\nabla} A, \nabla^2 A, \ldots) + g(A, \vec{\nabla} A, \nabla^2 A, \ldots)\xi(\vec{x}, t) \tag{11.1}$$

where we are going to assume that $\xi(x, t)$ is a Gaussian noise of zero mean and correlation

$$\left\langle \xi(\vec{x}, t)\xi(\vec{x}\,', t') \right\rangle = \delta(\vec{x}\,' - \vec{x})\delta(t - t') \tag{11.2}$$

which means that the noise fluctuations taking place at different times or at different spatial locations are uncorrelated. This is an idealization. In any real system, there will be a spatial scale at which fluctuations at different points will no longer be uncorrelated. Think on the Brownian particle in a fluid described in Chapter 6. The Brownian particle is not a field, but the fluctuations originating from collisions of the water molecules can be viewed as a field. At distances comparable to the size of a water molecule, the fluctuations are correlated.

As an example, consider the growth of a surface under the deposition of particles. Assume the particles arrive at different points of the surface in a random way. Ideally, one can consider that the number of particles arriving at a given location is uncorrelated with the number of particles arriving at another location. This idealization fails if one looks at spatial scales on the order of the size of the particle being deposited. However, typically these particles are molecules and one is interested in describing the macroscopic scales of the surface, which are much larger. In this situation, the assumption of uncorrelated arrival at different locations is justified.

11.1.1 Kardar-Parisi-Zhang Equation

In 1986, Mehran Kardar, Giorgio Parisi, and Yi-Cheng Zhang (KPZ) introduced the following stochastic PDE to model the temporal evolution of a growing interface:

$$\frac{\partial h(\vec{x}, t)}{\partial t} = \nu \nabla^2 h(\vec{x}, t) + \frac{\lambda}{2} \left|\nabla h(\vec{x}, t)\right|^2 + \sqrt{D}\xi(\vec{x}, t) \tag{11.3}$$

where $h(\vec{x}, t)$ is the height of the interface and $\xi(\vec{x}, t)$ is a Gaussian white noise of zero mean and correlations (11.2) and ν, λ and D are the parameters of the model. Here, the noise term, of intensity D, models the random arrival of particles at the interface, which is assumed to be uncorrelated at different points and at different times. The first term in the right-hand-side describes the diffusion along the surface which tends to smooth the surface. The parameter ν determines the strength of the diffusion. The second term of the right-hand-side describes the growth perpendicular to the interface. The parameter λ is associated to the deposition rate. The equation is written in the reference frame that moves with the

average surface (also known as comoving frame). In particular, for a system with one spatial dimension, the KPZ equation reads as

$$\frac{\partial h(x,t)}{\partial t} = \nu \frac{\partial^2 h(x,t)}{\partial x^2} + \frac{\lambda}{2}\left(\frac{\partial h(x,t)}{\partial x}\right)^2 + \sqrt{D}\xi(x,t). \tag{11.4}$$

In Sections 11.3 and 11.6, we will use the KPZ equation to illustrate the numerical methods for stochastic PDEs.

11.2 Coarse Graining

In order to derive a numerical method to tackle (11.1), we discretize the space in a regular lattice. For the sake of simplicity, let us consider, first, that we have only one spatial dimension x and that (11.1) describes the dynamics of the field in the spatial domain $[0, L]$.

The discretization process leads the division of this spatial domain in N cells of size $\Delta x = K/N$ (see Figure 11.1). For the deterministic PDE, one typically considers the value of the field at the discretization points. In a stochastic differential equation, the value of the noise at a particular point is a random number with infinite variance, and thus we have to proceed in a different way. As we did for the discretization in time of the stochastic differential equation in Chapter 6, a natural way out is to integrate. Therefore, instead of considering the values of the field at the specific discretization points, we consider a coarse-grain value which averages the value of the field over the entire cell. For the field A on cell n, one has

$$A_n(t) = \frac{1}{\Delta x}\int_{x_n}^{x_{n+1}} A(x,t)dx. \tag{11.5}$$

For the Gaussian white noise, the coarse-graining process leads to

$$\xi_n^{cg}(t) = \frac{1}{\Delta x}\int_{x_n}^{x_{n+1}} \xi(x,t)dx. \tag{11.6}$$

The stochastic process $\xi_n^{cg}(t)$ is Gaussian, with zero mean:

$$\left\langle \xi_n^{cg}(t)\right\rangle = \left\langle \frac{1}{\Delta x}\int_{x_n}^{x_{n+1}} \xi(x,t)dx\right\rangle = \frac{1}{\Delta x}\int_{x_n}^{x_{n+1}} \langle \xi(x,t)\rangle\, dx = 0 \tag{11.7}$$

and correlation given by

$$\left\langle \xi_n^{cg}(t)\xi_m^{cg}(t')\right\rangle = \frac{1}{(\Delta x)^2}\int_{x_n}^{x_{n+1}}\int_{x_m}^{x_{m+1}} \left\langle \xi(x,t)\xi(x',t')\right\rangle dxdx'. \tag{11.8}$$

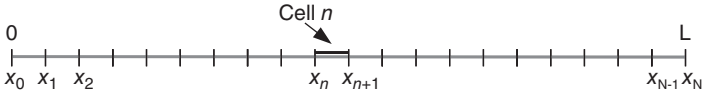


Figure 11.1 Discretization of the space in a regular lattice in one spatial dimension.

For $n \neq m$, the two integrals do not overlap and the correlation is zero. For $n = m$, one has

$$\frac{1}{(\Delta x)^2}\int_{x_n}^{x_{n+1}}\int_{x_n}^{x_{n+1}} \delta(x-x')\delta(t-t')dxdx' = \frac{1}{\Delta x}\delta(t-t'). \tag{11.9}$$

Therefore,

$$\left\langle \xi_n^{cg}(t)\xi_m^{cg}(t)\right\rangle = \frac{1}{\Delta x}\delta_{n,m}\delta(t-t') \tag{11.10}$$

that is, the correlation scales with the inverse of the cell size. As a consequence, the coarse-grained noise $\xi_n^{cg}(t)$ can be written as

$$\xi_n^{cg}(t) = \frac{1}{\sqrt{\Delta x}}\xi_n(t) \tag{11.11}$$

with $\left\langle \xi_n(t)\right\rangle = 0$ and $\left\langle \xi_n(t)\xi_m(t')\right\rangle = \delta_{nm}\delta(t-t')$.

We should emphasize the different scaling of the averaged field and the averaged noise with the cell size. In particular, the fact that the coarse-grained noise scales with the inverse of the square root of the cell size may seem awkward. However, it can be understood by comparing this with what we encountered in Chapter 7 when integrating the simple stochastic differential equation (7.2) over a small time interval h. In (7.6), $f_h(t)$ is proportional to h while $w_h(t)$ is of size $\sqrt{h}$. Translating this scaling to space, the integral over the cell of a smooth function is proportional to Δx, while the one of the noise is proportional to $\sqrt{\Delta x}$. Here, to obtain the coarse-grained variables, we are dealing with averages over the cell, so one has to divide by Δx. This leads to a coarse-grained field which scales as $(\Delta x)^0$ and a coarse-grained noise that scales as $1/\sqrt{\Delta x}$.

Furthermore, this scaling of the noise is consistent with the definition of the correlation function as a delta in space and with the continuous limit of a sum leading to an integral. To illustrate this, let us average the correlation function over the whole interval $[0, L]$. On the continuous, we have

$$\int_0^L\int_0^L \left\langle \xi(x,t)\xi(x',t')\right\rangle dxdx' = \int_0^L \delta(x-x')\delta(t-t')dx = L\delta(t-t'). \tag{11.12}$$

Let us now repeat the calculation on the discrete space. When discretizing, the integral is replaced by a sum and dx by Δx:

$$\begin{aligned}\sum_{n=1}^{N}\sum_{m=1}^{N}\Delta x\Delta x\left\langle (\Delta x)^{-1/2}\xi_n(t)(\Delta x)^{-1/2}\xi_m(t')\right\rangle &= \Delta x\sum_{n=1}^{N}\sum_{m=1}^{N}\delta_{nm}\delta(t-t') \\ &= \Delta x\sum_{n=1}^{N}\delta(t-t') = N\Delta x\delta(t-t')\,.\end{aligned} \tag{11.13}$$

Since $L = N\Delta x$, the result is the same.

For spatially extended systems with more than one spatial dimension, $\vec{x} = (x_1, \ldots, x_d)$, whose dynamics is described by an equation of the form (11.1), one can proceed in a similar way. Consider an hypercube of side L and divide this into small cells, which here are hypercubes of side Δx whose volume is $(\Delta x)^d$. Then,

one can average the value of the field over a given cell $\vec{n} = (n_1, \dots, n_d)$ to obtain $A_{\vec{n}}(t)$, which scales with $(\Delta x)^0$. For the noise, the coarse-graining leads to a $\xi^{cg}_{\vec{n}}(t)$, which can be written as

$$\xi^{cg}_{\vec{n}}(t) = \frac{1}{(\Delta x)^{d/2}} \xi_{\vec{n}}(t) \tag{11.14}$$

where $\xi_{\vec{n}}(t)$ are Gaussian white noises of zero mean and correlation $\left\langle \xi_{\vec{n}}(t)\xi_{\vec{m}}(t') \right\rangle = \delta_{\vec{n}\vec{m}}\delta(t-t')$.

This procedure can be easily generalized to systems that have different sizes in the different spatial dimensions, $\vec{L} = (L_1, \dots, L_d)$, for which one can discretize using a different number of cells for each spatial dimension, $\vec{N} = (N_1, \dots, N_d)$. Defining $\Delta\vec{x} = (\Delta x_1, \dots, \Delta x_d)$, where $\Delta_i = L_i/N_i$, the volume of the elementary cells is given by $V_c = \prod_{i=1}^{d} \Delta x_i$. The coarse-grained noise scales with $1/\sqrt{V_c}$.

11.3 Finite Difference Methods for Stochastic Differential Equations

Once the coarse-grained fields have been defined, the spatial derivatives are replaced by finite differences. This leads to a set of stochastic differential equations of the form (7.103) for the coarse-grained fields subject to the influence of the coarse-grained noises. The coupling terms in this set of ordinary differential equations originate from the spatial derivatives. There are several ways in which the spatial derivatives can be evaluated, and this will lead to a different set of ordinary differential equations to describe the same PDE (with, probably, a different degree of accuracy).

One way to discretize the spatial derivatives is the centered space method, in which they are evaluated using the coarse-grained values for the field in the neighboring cells in a symmetrical way. To illustrate this method, we consider a one-dimensional system. The first- and second-order spatial derivatives are evaluated as

$$\begin{aligned} \left.\frac{\partial A}{\partial x}\right|_{x_n,t_i} &\to \frac{A_{n+1}(t_i) - A_{n-1}(t_i)}{2\Delta x} + O[(\Delta x)^2] \\ \left.\frac{\partial^2 A}{\partial x^2}\right|_{x_n,t_i} &\to \frac{A_{n+1}(t_i) - 2A_n(t_i) + A_{n-1}(t_i)}{(\Delta x)^2} + O[(\Delta x)^2]. \end{aligned} \tag{11.15}$$

In general, for the derivatives of any order, the error in the spatial discretization is $O[(\Delta x)^2]$, so the centered space method is said to be of second order.

Once the spatial derivatives have been transformed to finite differences, we have a set of coupled ordinary differential equations of the form (7.103) for the set of variables $(A_0, \dots, A_{N-1})$ subject to the noises $(\xi_0(t), \dots, \xi_{N-1}(t))$. The amplitude scale for the noise $1/\sqrt{\Delta x}$ has to be included in the noise terms g_{nm}, which are now function of $(A_0, \dots, A_{N-1})$. In fact, if the equation for this one dimensional field is of the form (11.1), the noise in cell n will appear only in the equation for A_n, and therefore the diffusion terms g_{nm} will be of the form $g_{nm} = g_n\delta_{nm}$. Technically, this is called a *stochastic PDE with a single-point multiplicative noise*.

As an example, let us consider the spatial discretization of the KPZ equation in one spatial dimension (11.4). One has

$$\frac{dh_n(t)}{dt} = \nu \frac{h_{n+1}(t) - 2h_n(t) + h_{n-1}(t)}{(\Delta x)^2} + \frac{\lambda}{2}\left(\frac{h_{n+1}(t) - h_{n-1}(t)}{2\Delta x}\right)^2 + \frac{1}{\sqrt{\Delta x}}\xi_n(t). \tag{11.16}$$

If the stochastic PDE includes a noise term of the form

$$\frac{\partial^j A(x,t)\xi(x,t)}{\partial x^j} \tag{11.17}$$

or an integral term of the form

$$\int_0^L g(A(x,t))\xi(x,t)dx \tag{11.18}$$

then after discretization one obtains that dynamics of the coarse-grained field at cell n depends on

$$\sum_{m=1}^{N} g_{nm}(A_0, \ldots, A_{N-1})\xi_m(t) \tag{11.19}$$

where, typically, g_{nm} will have also off-diagonal contributions. For (11.17), g_{nm} is nonzero only when the locations n, m are nearest-neighbors in the regular lattice. This leads to a sparse matrix in which the nonzero terms are located in the main diagonal and one or more diagonals on either side: that is, it is a band-diagonal matrix. On the contrary, for a noise term of the form (11.18), g_{nm} is typically a dense matrix. Both (11.17) and (11.18) are examples of multiple-point multiplicative noises.

The centered space method can be extended to a system in d spatial dimensions. For instance, the first derivative along dimension j is given by

$$\left.\frac{\partial A}{\partial x_i}\right|_{x_{\vec{n}}, t_i} \to \frac{1}{2\Delta x_i}\left[A_{\vec{n}_i^+}(t_i) - A_{\vec{n}_i^-}(t_i)\right] + O[(\Delta x_i)^2] \tag{11.20}$$

where $\vec{n}_i^{\pm} = (n_1, \ldots, n_{i-1}, n_i \pm 1, n_{i+1}, \ldots, n_d)$. Higher order derivatives are defined in a similar way. Again, once the spatial derivatives have been transformed into finite differences, we get a set of coupled stochastic differential equations of the form (7.103) for the variables $A_{\vec{n}}$ driven by the noises $\xi_{\vec{m}}$. Then these equations can be integrated using, for instance, the Milshtein or the Heun method as discussed in Section 7.5. However, the fact that the coupling between the equations originates from spatial derivatives will have consequences on the numerical stability of these equations, as we will discuss in Section 11.4.

We also note that, in order to properly model the system, it is also necessary that the coarse-grained fields at the boundary cells obey the boundary conditions. For instance, for the discretization of the KPZ equation given by (11.16), one would have to specify how the derivatives are evaluated at the borders, since A_{-1} and A_N are not defined. Here, we will be mainly interested in the dynamics at the bulk, far away from any boundary; in these cases one can avoid the boundary effect by considering a sufficiently large system[1)] with periodic boundary conditions.

1) By sufficiently large we mean a system whose size is much larger than the typical space scale of the dynamics.

Centered derivatives are not always the best option. If the system has a preferred direction for the propagation of the field, for instance, there is a flux in a given direction or the dynamics includes an advection term, then a centered space method, which is symmetric, may not be suitable since it does not preserve the natural asymmetry of the system. In this situation, it is preferable to evaluate the spatial derivatives in an upstream way, namely to use only spatial points that are allowed to contribute to the dynamics according to the asymmetries of the original PDE. As an example, consider a one-dimensional system in which the field moves to the left; then we discretize the spatial derivative as

$$\left.\frac{\partial A}{\partial x}\right|_{x_n,t_i} \to \frac{A_{n+1}(t_i) - A_n(t_i)}{\Delta x} + O[(\Delta x)] \tag{11.21}$$

while if the field moves to the right we use

$$\left.\frac{\partial A}{\partial x}\right|_{x_n,t_i} \to \frac{A_n(t_i) - A_{n-1}(t_i)}{\Delta x} + O[(\Delta x)]. \tag{11.22}$$

Note that the discretization error in (11.21) and (11.22) is of order 1 while it was of order 2 for the symmetric discretization (11.15). However, the most important point for any numerical simulation is to reproduce the fundamental properties of the original system, and symmetry is one of these. Thus, for a system where the field moves in a preferred direction, upstream derivatives will, in general, provide better results, despite using only first-order derivatives. For systems with several space dimensions in which in there is a flux in a specific direction in one of the dimensions, one can combine upstream derivatives in that dimension and centered derivatives in the others.

11.4
Time Discretization: von Neumann Stability Analysis

After the discretization in space, we have converted the stochastic PDE into a set of coupled stochastic differential equations of the form (7.103). As discussed in Section 11.3, the coupling between the variables $A_{\vec{n}}(t)$ comes from the spatial derivatives. The discretization process introduces a space scale which was not present in the stochastic PDE. We will show now that the presence of this space scale, together with the form of the coupling, has profound implications in the time discretization. As a result, some combinations of space and temporal discretizations are numerically unstable when applied to certain kinds of PDEs. In other instances, the space and temporal discretizations lead to a stable algorithm provided the time step is smaller than a certain maximum value, which in practice means that you have to be careful in setting the time step.

Assume that you discretize the time so that, finally, the set of equations of the form (7.103) originated from a stochastic PDE is converted to a set of coupled maps, namely a set of coupled difference equations, for the variables $A_{\vec{n}}(t_i)$ with $t_i = ih$. For the moment, we are going to neglect the noise terms and consider only

the deterministic part of the map. The idea of von Neumann stability analysis is to test the stability of this map, which is typically nonlinear, against numerical perturbations. If perturbations grow, the map will generate numerical divergences and the discretization method you have used is clearly not suitable for that equation. However, the conversely is not true: the absence of numerical instabilities does not mean that the method is a good one or even a suitable one. We just know that it will not diverge, but it does not ensure accuracy. Besides, in this analysis we are not testing the accuracy of the stochastic part.

The von Neumann stability analysis can be performed as follows: Consider that we have a solution of the deterministic coupled map A^s and we add a small perturbation to it:

$$A_{\vec{n}}(t_i) = A^s_{\vec{n}}(t_i) + \epsilon a_{\vec{n}}(t_i) \tag{11.23}$$

where ϵ determines the overall size of the perturbation (considered small). One substitutes (11.23) into the set of coupled maps and expands in power series of ϵ. The zero-order term satisfies the set of coupled maps. We keep the terms at order ϵ and disregard the higher order terms. This leads to a set of linearly coupled maps, which, assuming periodic boundary conditions, can be solved with the ansatz

$$a_{\vec{n}}(t_i) = \lambda(\vec{k})^i e^{i\vec{k}\vec{n}\Delta x} \tag{11.24}$$

where $\lambda(\vec{k})$ is a complex number. Basically, this corresponds to the discrete Fourier transformation in space of the set of values $A_{\vec{n}}(t_i)$. We discuss in more detail the discrete Fourier transform in Appendix G. Here we will just take (11.24) as an ansatz which is useful to solve the set of linear maps because the coupling between the variables originates from spatial derivatives. The amplitude of each Fourier mode at integration time step i is given by $\lambda(k)^i$. Thus if $|\lambda(k)| < 1$ (or equivalently if $|\lambda(k)|^2 < 1$) for all k, the set of coupled maps is stable. If for any k, $|\lambda(k)| > 1$, the set of coupled maps is unstable.

In case the field is a vector with d components $\vec{A} = (A_1, \dots, A_d)$ whose dynamics is given by

$$\frac{\partial A_j(\vec{x}, t)}{\partial t} = f_j(\vec{A}(\vec{x}, t)) \tag{11.25}$$

then one can proceed in a similar way. Once the system is discretized, one assumes that $\vec{A}^s$ is a solution of the set of coupled maps and adds a perturbation of the form

$$A_{j\vec{n}}(t_i) = A^s_{j\vec{n}}(t_i) + \epsilon a_{j\vec{n}}(t_i). \tag{11.26}$$

Substituting this last expression into the set of nonlinear maps and expanding to the first order in ϵ one gets a set of linear maps for $a_{j\vec{n}}(t_j)$. Then one looks for solutions of the form

$$a_{j\vec{n}}(t_i) = \lambda(\vec{k})^i e^{i\vec{k}\vec{n}\Delta x} a^0_j \tag{11.27}$$

which leads to a set of equations for $\lambda(\vec{k})$.

We now consider the application of the von Neumann analysis to the deterministic part of a set of equations of the form (7.103) originating from a stochastic PDE,

which can be integrated, for instance, using the Milshtein or the Heun algorithms. We first consider the implications for the Milshtein algorithm in which the deterministic part the time evolution is implemented as in the forward Euler method.

We start by considering the diffusion equation

$$\frac{\partial A(x,t)}{\partial t} = D\frac{\partial^2 A(x,t)}{\partial x^2}. \tag{11.28}$$

We discretize the spatial derivative following the centered space method and the temporal derivatives using the forward Euler method. This combination is known as the *forward time centered space method* for deterministic PDEs. One has

$$\frac{A_n(t_{i+1}) - A_n(t_i)}{h} = D\frac{A_{n+1}(t_i) - 2A_n(t_i) + A_{n-1}(t_i)}{(\Delta x)^2}. \tag{11.29}$$

One can explicitly write the value of the field at time t_{i+1} as a function of the values of the field at time t:

$$A_n(t_{i+1}) = A_n(t_i) + \frac{Dh}{(\Delta x)^2}\left[A_{n+1}(t_i) - 2A_n(t_i) + A_{n-1}(t_i)\right]. \tag{11.30}$$

We now apply the von Neumann stability analysis by considering perturbations of the form (11.23). Since the set of maps (11.30) is linear, the perturbations $a_n(t_i)$ fulfill the same set of maps (11.30), namely

$$a_n(t_{i+1}) = a_n(t_i) + \frac{Dh}{(\Delta x)^2}\left[a_{n+1}(t_i) - 2a_n(t_i) + a_{n-1}(t_i)\right]. \tag{11.31}$$

Applying the von Neumann ansatz (11.24) one has

$$\begin{aligned}\lambda(k)^{i+1}\mathrm{e}^{\mathrm{i}kn\Delta x} &= \lambda(k)^i\mathrm{e}^{\mathrm{i}kn\Delta x} \\ &\quad + \frac{Dh}{(\Delta x)^2}\left[\lambda(k)^i\mathrm{e}^{\mathrm{i}k(n+1)\Delta x} - 2\lambda(k)^i\mathrm{e}^{\mathrm{i}kn\Delta x} + \lambda(k)^i\mathrm{e}^{\mathrm{i}k(n-1)\Delta x}\right].\end{aligned} \tag{11.32}$$

Dividing by $\lambda(k)^i\mathrm{e}^{\mathrm{i}kn\Delta x}$, one gets

$$\lambda(k) = 1 + \frac{Dh}{(\Delta x)^2}\left[\mathrm{e}^{\mathrm{i}k\Delta x} - 2 + \mathrm{e}^{-\mathrm{i}k\Delta x}\right] = 1 - \frac{4Dh}{(\Delta x)^2}\sin^2(k\Delta x/2). \tag{11.33}$$

In order for the method to be stable, it is required that $|\lambda(k)| < 1$ for all k. Note that in our case $\lambda(k)$ can never be larger than 1, but it can be smaller than -1. The worst case scenario is given by $k\Delta x = \pi$ for which $\lambda(\pi/\Delta x) = 1 - 4Dh/(\Delta x)^2$. Imposing $1 - 4Dh/(\Delta x)^2 \geq -1$ leads to

$$\frac{2Dh}{(\Delta x)^2} \leq 1. \tag{11.34}$$

This stability condition, known as the *Courant criterion* or the *Courant–Friedrichs–Lewy criterion*, has an interesting physical interpretation. To evaluate the field at $x = x_n$ at time $t_i + h$, we have used the values of the field at time t_i at the nearest neighbor points $x = x_{n\pm1}$. If h is small enough, this is fine since whatever happens at time t_i at spatial points located further away has no time to affect the field at location $x = x_n$ at time $t_i + h$. However, if h is too large, then, in order to properly evaluate the field at location $x = x_n$ at time $t_i + h$, one needs to consider the value of the field at time t_i at spatial points beyond the next neighbors. In other words,

the Courant criterion implies that the numerical domain of dependence (namely the set of lattice points used to evaluate the field at the next time step at a given location) must contain the physical domain of dependence (the region of space that can influence the value of the field at that location at the next time step according to the original PDE).

The practical consequence of the Courant criterion for the forward time centered space method applied to the diffusion equation is that, if one increases the spatial resolution by a factor 2, it is forced to take a time step 4 times smaller, and so the computer time will increase roughly by a factor 8.

Let us move one step further and consider that the PDE contains first- and second-order spatial derivatives, such as the Fokker–Planck equation with constant coefficients:

$$\frac{\partial f(x;t)}{\partial t} = B\frac{\partial f(x;t)}{\partial x} + D\frac{\partial^2 f(x;t)}{\partial x^2}. \tag{11.35}$$

Using the forward time centered space method to discretize (11.35) leads to

$$\begin{aligned} f(x_n;t_{i+1}) = f(x_n;t_i) &+ \frac{Bh}{2\Delta x}\left[f(x_{n+1};t_i) - f(x_{n-1};t_i)\right] \\ &+ \frac{Dh}{(\Delta x)^2}\left[f(x_{n+1};t_i) - 2f(x_n;t_i) + f(x_{n-1};t_i)\right]. \end{aligned} \tag{11.36}$$

We can now apply the von Neumann stability analysis (11.23)–(11.24) to (11.36). After dividing by $\lambda(k)^i e^{\mathbf{i}kn\Delta x}$, one has

$$\begin{aligned} \lambda(k) &= 1 + \frac{Bh}{2\Delta x}\left(\mathrm{e}^{\mathbf{i}k\Delta x} - \mathrm{e}^{-\mathbf{i}k\Delta x}\right) + \frac{Dh}{(\Delta x)^2}\left[\mathrm{e}^{\mathbf{i}k\Delta x} - 2 + \mathrm{e}^{-\mathbf{i}k\Delta x}\right] \\ &= 1 + \mathbf{i}\frac{ah}{\Delta x}\sin(k\Delta x) - \frac{4Dh}{(\Delta x)^2}\sin^2(k\Delta x/2). \end{aligned} \tag{11.37}$$

The absolute value of $\lambda(k)$ is given by

$$|\lambda(k)|^2 = \left[1 - \frac{4Dh}{(\Delta x)^2}\sin^2(k\Delta x/2)\right]^2 + \left[\frac{Bh}{\Delta x}\sin(k\Delta x)\right]^2. \tag{11.38}$$

The first thing to notice is that, since both terms are positive, a necessary condition for the stability is that both are smaller than 1, namely

$$\frac{2Dh}{(\Delta x)^2} \leq 1\,, \quad \frac{Bh}{\Delta x} \leq 1. \tag{11.39}$$

However, this is not sufficient, since each term can be smaller than 1 and yet the sum be larger than 1.

A second remark is that, in the absence of diffusion, $D = 0$, $|\lambda(k)| \geq 1$ for all k, and therefore the forward time centered space method is unconditionally unstable for an equation of the form

$$\frac{\partial A(x,t)}{\partial t} = -v\frac{\partial A(x,t)}{\partial x} \tag{11.40}$$

which describes the advection of the field A along the x-axis with velocity v. Thus a discretization scheme suitable for PDEs with both first- and second-order spatial derivatives can be unstable, leading to numerical divergences, for a simpler PDE

with only first-order spatial derivatives. While this may seem counterintuitive, it has a physical explanation. Equation (11.40) can be solved exactly, $A(x,t) = A(x - vt, 0)$. In other words, the field can have an arbitrary profile set by the initial condition and this profile moves rigidly at a constant velocity v. Therefore, this is somehow a tricky situation in which perturbations should neither decay nor grow. The forward time centered space method cannot ensure the zero growth rate of the perturbations; instead, it provides a net positive growth that leads to divergences. On the contrary, in the Fokker–Planck equation (11.35) diffusion broadens the profile of the field. The physical diffusion stabilizes the numerical method, provided D is large enough and h and Δx are chosen so that $|\lambda(k)| \leq 1$.

It can be shown, see Exercise 1, that for (11.40) the upstream spatial derivative method is stable provided that

$$\frac{|v|h}{\Delta x} \leq 1 \tag{11.41}$$

which is the Courant–Friedrichs–Lewy criterion for the advection equation and whose physical interpretation is the same as before. The size of the time steps is limited by the size of the spatial domain used in the numerical evaluation, and, as before, the physical spatial domain must be inside the numerical domain for stability. The difference comes from the fact that now perturbations propagate at speed v instead of diffusing, and thus the propagation distance grows linearly with time. In fact, we have already encountered this criterion as one of the necessary conditions for the stability of the forward time centered space method for the Fokker–Planck equation (11.39). For the upstream discretization with $h < \Delta x/|v|$, all perturbations except those with wave number $k = 0$ decay, which is not the case for the advection PDE (11.40). The fact that perturbations decay is good for numerical stability but it also means that numerically any initial arbitrary profile will be damped instead of being propagated without deformation as predicted by the continuous equation. The key point is that not all perturbations decay at the same rate. Typically, one is interested in the dynamics of the system at space scales that are much larger than the spatial discretization: that is, one is interested in the dynamics at small wave numbers k and, as shown in Exercise 1, small wave number perturbations are only weakly damped. Provided one is interested in the dynamics at large spatial scales and over time intervals in which the damping is not noticeable, this method can be used to integrate the advection equation.

Let us now consider the KPZ equation. Discretizing the time, the Milshtein method for the set of stochastic differential equations (11.16) leads to

$$\begin{aligned} h_n(t_{i+1}) = h_n(t_i) &+ \frac{\nu h}{(\Delta x)^2}\left[h_{n+1}(t_i) - 2h_n(t_i) + h_{n-1}(t_i)\right] \\ &+ \frac{\lambda h}{8(\Delta x)^2}\left[h_{n+1}(t_i) - h_{n-1}(t_i)\right]^2 + \frac{\sqrt{Dh}}{\sqrt{\Delta x}}u_{ni}, \end{aligned} \tag{11.42}$$

where $\{u_{ni}\}$ is a set of uncorrelated Gaussian random numbers of zero mean and variance 1.

We now proceed to perform the von Neumann stability analysis. The deterministic part is a set of coupled nonlinear maps, so we introduce $h_n(t_i) = h_n^s(t_i) + \epsilon a_n(t_i)$

assuming $h_n^s(t_i)$ is a solution of (11.42). Expanding up to first order in ϵ, one has

$$a_n(t_{i+1}) = a_n(t_i) + \frac{\nu h}{(\Delta x)^2}\left[a_{n+1}(t_i) - 2a_n(t_i) + a_{n-1}(t_i)\right] + \frac{\lambda h}{2(\Delta x)}\frac{h_{n+1}^s(t_i) - h_{n-1}^s(t_i)}{2\Delta x}\left[a_{n+1}(t_i) - a_{n-1}(t_i)\right]. \tag{11.43}$$

Applying he von Neumann ansatz results in

$$\lambda(k) = 1 - \frac{4\nu h}{(\Delta x)^2}\sin^2\left(\frac{k\Delta x}{2}\right) + \mathbf{i}\frac{\lambda h}{\Delta x}\frac{h_{n+1}^s(t_i) - h_{n-1}^s(t_i)}{2\Delta x}\sin(k\Delta x). \tag{11.44}$$

Now the value of $\lambda(k)$ depends on the exact solution $h_n^s(t_i)$, which we do not know *a priori*. The absolute value squared is given by

$$|\lambda(k)|^2 = \left[1 - \frac{4\nu h}{(\Delta x)^2}\sin^2\left(\frac{k\Delta x}{2}\right)\right]^2 + \left[\frac{\lambda h}{\Delta x}\frac{h_{n+1}^s(t_i) - h_{n-1}^s(t_i)}{2\Delta x}\sin(k\Delta x)\right]^2. \tag{11.45}$$

Even if we do not know the values of $h_n^s(t_i)$, (11.45) implies that the Courant–Friedrichs–Lewy criterion $\nu h/(\Delta x)^2 < 1$ is a necessary (but not sufficient) condition for numerical stability. Furthermore, assuming that the absolute value of discrete derivative is bounded, that is

$$\left|\frac{h_{n+1}^s(t_i) - h_{n-1}^s(t_i)}{2\Delta x}\right| < K \tag{11.46}$$

one obtains

$$|\lambda(k)|^2 < \left[1 - \frac{4\nu h}{(\Delta x)^2}\sin^2\left(\frac{k\Delta x}{2}\right)\right]^2 + \left[\frac{\lambda K h}{\Delta x}\sin(k\Delta x)\right]^2 \tag{11.47}$$

which has the same form as (11.38). So numerical stability of the deterministic part of the KPZ requires the diffusion coefficient ν to be large enough as discussed above.

So far, we have considered that the temporal discretization is performed using the explicit Euler method. Let us now take a look at the Heun method, in which the time derivative is also explicit but is of second order. Let us see whether the additional order allows us to overcome the time-step limitations imposed by the Courant–Friedrichs–Lewy criterion. For instance, consider the diffusion equation (11.28), for which the Heun method reads

$$\begin{aligned} A_n^{(1)} &= A_n(t_i) + \frac{Dh}{(\Delta x)^2}\left[A_{n+1}(t_i) - 2A_n(t_i) + A_{n-1}(t_i)\right] \\ A_n(t_{i+1}) &= A_n(t_i) + \frac{Dh}{2(\Delta x)^2}\Big[A_{n+1}(t_i) - 2A_n(t_i) + A_{n-1}(t_i) \\ &\quad + A_{n+1}^{(1)} - 2A_n^{(1)} + A_{n-1}^{(1)}\Big]. \end{aligned} \tag{11.48}$$

The von Neumann stability analysis (see Exercise 2) leads to

$$\lambda(k) = 1 - \frac{4Dh}{(\Delta x)^2}\sin^2\left(\frac{k\Delta x}{2}\right) + \frac{8D^2h^2}{(\Delta x)^4}\sin^4\left(\frac{k\Delta x}{2}\right). \tag{11.49}$$

Imposing $|\lambda(k)| < 1$ leads exactly to the same the Courant–Friedrichs–Lewy criterion as before (11.34) (see Exercise 2), so for the diffusion equation the stability is the same as using the Euler method for time discretization.

The Heun method for the KPZ equation reads

$$
\begin{aligned}
h_n^{(1)} &= h_n(t_i) + \frac{\nu h}{(\Delta x)^2}\left[h_{n+1}(t_i) - 2h_n(t_i) + h_{n-1}(t_i)\right] \\
&\quad + \frac{\lambda h}{8(\Delta x)^2}\left[h_{n+1}(t_i) - h_{n-1}(t_i)\right]^2 + \frac{\sqrt{Dh}}{\sqrt{\Delta x}}u_{ni} \\
h_n(t_{i+1}) &= h_n(t_i) + \frac{\nu h}{2(\Delta x)^2}\left[h_{n+1}(t_i) - 2h_n(t_i) + h_{n-1}(t_i) + h_{n+1}^{(1)} - 2h_n^{(1)} + h_{n-1}^{(1)}\right] \\
&\quad + \frac{\lambda h}{16(\Delta x)^2}\left[\left[h_{n+1}(t_i) - h_{n-1}(t_i)\right]^2 + \left[h_{n+1}^{(1)} - h_{n-1}^{(1)}\right]^2\right] \\
&\quad + \frac{\sqrt{Dh}}{\sqrt{\Delta x}}u_{ni}.
\end{aligned}
\tag{11.50}
$$

One can show (see Exercise 3) that the Courant–Friedrichs–Lewy criterion (11.34) is a necessary condition for stability of the deterministic part of this map.

It turns out that the Courant–Friedrichs–Lewy criterion is quite general (11.34) in the sense that any typical finite differences method with explicit temporal derivatives applied to the diffusion equation (11.28) requires $h < (\Delta x)^2/2D$ to be numerically stable. The same criterion appeared as one of the necessary conditions for the KPZ equation which also has second-order spatial derivatives but is nonlinear. Experience indicates that nonlinearities usually do not cure the instabilities generated by a wrong spatiotemporal discretization; on the contrary, typically the dynamics can be more complex requiring a higher spatial and temporal resolution. Therefore, when using explicit finite differences methods to integrate PDEs with second-order spatial derivatives, one has to be aware that the maximum time step scales with $(\Delta x)^2$. Similarly for first-order spatial derivatives the Courant-Friedrichs-Lewy criterium (11.41) applies and the maximum time step scales with Δx.

The situation becomes worse for higher order derivatives. In general, if the PDE includes a spatial derivative of order n, the maximum time step for explicit methods scales with the inverse of the spatial step at the power n, namely $h_{\max} \propto (\Delta x)^n$.

Implicit or semi-implicit methods are usually more stable. For instance, the semi-implicit Crank–Nicolson method (see Exercise 4) is unconditionally stable for the diffusion equation. However, implicit and semi-implicit methods are quite cumbersome to implement and stability does not imply accuracy.

We finish this section by providing examples of the implementation of the Milshtein and Heun finite differences methods for the KPZ equation. Here we consider periodic boundary conditions. The core of the Milshtein method implementing one time step can be written as

```
hl=cshift(hn,-1)
hr=cshift(hn,1)
hn=hn+diff*(hl-2.d0*hn+hr)+growth*(hl-hr)**2+eqs*dran_g()
t=t+h
```

where `hn` is a real (double precision) vector of length N, which at the beginning of the step contains $h_n(t_i)$ and at the end stores $h_n(t_{i+1})$ to be used in the next time step. In what follows, when referring to computer programs by real we mean a real*8

or double precision number. We have used the Fortran90 instruction cshift which performs a circular shift of the elements of an array. The real vector `hl` stores the elements of `hn` shifted one position to the left. Therefore `hl(n)=hn(n+1)`. Similarly, `hr` stores the elements of `hn` shifted one position to the right, so that `hr(n)=hn(n-1)`. The real number `diff` corresponds to $\nu h/(\Delta x)^2$, `growth` to $\lambda h/8(\Delta x)^2$, and `eqs` to $\sqrt{Dh/\Delta x}$.

The core of the Heun method can be programmed as follows:

```
hl=cshift(hn,-1)
hr=cshift(hn,1)
g=eqs*dran_g()
aux=diff*(hl-2.d0*hn+hr)+growth*(hl-hr)**2
h_1=hn+aux+g
hl=cshift(h_1,-1)
hr=cshift(h_1,1)
aux1=diff*(hl-2.d0*h_1+hr)+growth*(hl-hr)**2
hn=hn+0.5d0*(aux+aux1)+g
t=t+h
```

where the real vector `h_1` corresponds to $h_n^{(1)}$. The vectors `aux` and `aux1` are used to store temporary values, while the vector `g` stores the noise. The other variables have the same meaning as before.

11.5 Pseudospectral Algorithms for Deterministic Partial Differential Equations

There are other kinds of numerical methods that are useful when the boundary conditions have a regular geometry and satisfy some particular conditions. Those are the pseudospectral methods. The idea behind pseudospectral methods is to decompose the field into a set of basis functions (also referred to as ''modes'') that satisfy the boundary conditions. For instance, in systems with periodic boundary conditions, one can decompose the field in Fourier modes. Although in principle one can use pseudospectral methods with a variety of mode decompositions, in practice the most widely used pseudospectral methods are those that make use of Fourier decomposition, the reason being that they allow for the exact calculation of the spatial derivatives. An added advantage is the existence of the fast Fourier transform which allows a very efficient way to numerically calculate the Fourier integrals.

We will focus here on algorithms that make use of the decomposition of the field into Fourier modes. For pedagogical reasons, we will first introduce these algorithms for deterministic PDEs. The application of pseudospectral modes to stochastic differential equations will be discussed in Section 11.6.

It is convenient to separate the linear terms and the nonlinear terms of the PDE because these methods allow them to be treated in a different way. The basic idea, we will be more specific in a moment, is that we will integrate the linear terms exactly. So, we start writing the PDE in the form

$$\frac{\partial A(\vec{x},t)}{\partial t} = \mathcal{L}[A(\vec{x},t)] + \mathcal{N}[A(\vec{x},t)]. \tag{11.51}$$

We assume that the field $A(\vec{x},t)$ satisfies periodic boundary conditions in the interval $\vec{x} \in [0,L]^d$.

In general, we assume that the Fourier transform of $A(\vec{x},t)$, as defined in Appendix G, does exist, that is

$$\tilde{A}(\vec{q},t) \equiv \mathcal{F}_P[A(\vec{x},t)] = \frac{1}{L^d}\int_{[0,L]^d} A(\vec{x},t)\mathrm{e}^{\mathrm{i}\vec{q}\vec{x}}d\vec{x}. \tag{11.52}$$

The inverse transform is given by

$$A(\vec{x},t) \equiv \mathcal{F}_P^{-1}[\tilde{A}(\vec{q},t)] = \sum_{\vec{k}=-\infty}^{\infty} \tilde{A}(\vec{q}_{\vec{k}},t)\mathrm{e}^{-\mathrm{i}\frac{2\pi}{L}\vec{k}\vec{x}} \tag{11.53}$$

where, as in Appendix G, we use the notation $\vec{q}_{\vec{k}}(t) \equiv \frac{2\pi}{L}\vec{k}$, with $\vec{k} = (k_1,\ldots,k_d)$, and k_i being an integer number. We apply now this Fourier operator $\mathcal{F}_P$ to the differential equation (11.51). For example, integration by parts yields

$$\mathcal{F}_P\left[\frac{\partial A(x,t)}{\partial \vec{x}}\right]\Bigg|_{\vec{q}=\vec{q}_{\vec{k}}} = -\mathrm{i}\vec{q}_{\vec{k}}\tilde{A}(\vec{q}_{\vec{k}},t) \tag{11.54}$$

and therefore

$$\frac{\partial A(x,t)}{\partial \vec{x}} = -\mathcal{F}_P^{-1}[\mathrm{i}\vec{q}\tilde{A}(\vec{q},t)]. \tag{11.55}$$

As a consequence, under the Fourier transformation the linear term of the equation usually assumes an algebraic form.

In order to illustrate this methodology, we will consider a specific system in a one-dimensional space, namely the Nikolaevskii equation, which was originally proposed to describe the propagation of longitudinal seismic waves in viscoelastic media:

$$\frac{\partial u(x,t)}{\partial t} = -\frac{\partial^2}{\partial x^2}\left[\epsilon - \left(1+\frac{\partial^2}{\partial x^2}\right)^2\right]u(x,t) - \left(\frac{\partial u(x,t)}{\partial x}\right)^2. \tag{11.56}$$

From a numerical point of view, the major difficulty in the integration of this equation is that it relies on the higher order of the spatial derivatives (up to order 6). There are obvious difficulties associated with the calculation of these higher order derivatives with enough precision using finite differences. What is worse is that, for explicit finite difference methods, the Courant–Friedrichs–Lewy stability criterion would require that the integration time step scales with the spatial discretization step to the power 6 in order to avoid numerical instabilities. Thus reducing the spatial step by a factor 2 implies that one has to take a time step 64 times smaller, and therefore the computational time would be 128 times larger.

The linear part of (11.56) is given by

$$\mathcal{L}[u(x,t)] = -\frac{\partial^2}{\partial x^2}\left[\epsilon - \left(1+\frac{\partial^2}{\partial x^2}\right)^2\right]u(x,t) \tag{11.57}$$

while the nonlinear part is given by

$$\mathcal{N}[u(x,t)] = -\left(\frac{\partial u(x,t)}{\partial x}\right)^2. \tag{11.58}$$

The linear operator of the Nikolaevskii equation in Fourier space is

$$\mathcal{F}_P\left[-\frac{\partial^2}{\partial x^2}\left[\epsilon - \left(1+\frac{\partial^2}{\partial x^2}\right)^2\right]u(x,t)\right]\Bigg|_{q=q_k} = \omega(q_k)\tilde{u}(q_k,t) \tag{11.59}$$

where the dispersion relation $\omega(q)$ is given by

$$\omega(q) = q^2(\epsilon - (1-q^2)^2). \tag{11.60}$$

The nonlinear terms can also be written in Fourier space, but in a more complicated form. For instance, for the Nikolaevskii equation, we use

$$\left(\frac{\partial u(x,t)}{\partial x}\right)^2 = \left(\mathcal{F}_P^{-1}[\mathbf{i}q\tilde{u}(q,t)]\right)^2. \tag{11.61}$$

Then the Fourier transform of the nonlinear term can be written as

$$\mathcal{F}_P\left[\mathcal{N}[u(x,t)]\right] = -\mathcal{F}_P\left[\left(\mathcal{F}_P^{-1}[\mathbf{i}q\tilde{u}(q,t)]\right)^2\right]\Bigg|_{q=q_k} \equiv \tilde{a}(q_k,t). \tag{11.62}$$

Eq. (11.62) indicates that one goes from Fourier space to real space, evaluates the nonlinear term in real space and finally the result is transformed back to Fourier space. Finally, applying the Fourier transform to (11.56) leads to

$$\frac{\partial\tilde{u}(q_k,t)}{\partial t} = \omega(q_k)\tilde{u}(q_k,t) + \tilde{a}(q_k,t). \tag{11.63}$$

We approximate the coefficients $\tilde{u}_k$ of the Fourier series by the corresponding ones $\hat{u}_k$ of the discrete Fourier series using (G.5). The result is a set of N coupled (complex) equations for the discrete Fourier variables:

$$\frac{d\hat{u}_k(t)}{dt} = \omega_k\hat{u}_k(t) + \hat{a}_k(t), \qquad k = -\frac{N}{2}+1,\ldots,\frac{N}{2} \tag{11.64}$$

where

$$\omega_k = \omega(q_k), \tag{11.65}$$

$$\hat{u}_k(t) = \mathcal{F}_D[u(x,t)], \tag{11.66}$$

$$\hat{a}_k(t) = \mathcal{F}_D[\mathcal{N}[u(x,t)]]. \tag{11.67}$$

The equations for other values of k are not needed because, due to aliasing, they simply reproduce this basic set of equations. As shown in Appendix G, in the discrete Fourier space the spatial derivative corresponds to multiplying the function by $\mathbf{i}q_k$, where (see (18.25))

$$\begin{aligned} q_k &= \frac{2\pi}{L}k, \qquad k = -\frac{N}{2}+1,\ldots,\frac{N}{2}-1 \\ q_{\frac{N}{2}} &= 0. \end{aligned} \tag{11.68}$$

Therefore, in the discrete Fourier space, ω_k should be evaluated using (11.60) and (11.68).

While (11.64) has been obtained for the Nikolaevskii equation, its overall form is quite general. For a general PDE (11.51), one obtains a similar set of ordinary differential equations for the set of variables $\hat{A}_{\vec{k}}(t)$:

$$\frac{d\hat{A}_{\vec{k}}(t)}{dt} = \omega_{\vec{k}}\hat{A}_{\vec{k}}(t) + \hat{a}_{\vec{k}}(t) \tag{11.69}$$

where $\omega_{\vec{k}} = \omega(\vec{q}_{\vec{k}})$, $\hat{A}_{\vec{k}}(t) = \mathcal{F}_D[A(\vec{x}, t)]$, and $\hat{a}_{\vec{k}} = \mathcal{F}_D[\mathcal{N}[A(\vec{x}, t)]]$.

The main idea behind the pseudospectral methods we will discuss below is to do the integration in the discrete Fourier space using a set of ordinary differential equations of the form (11.64) for a one-dimensional system or of the form (11.69) for a system with several spatial dimensions. The key point is that the Fourier modes $\hat{u}_k(t)$, or in general $\hat{A}_{\vec{k}}(t)$, are linearly decoupled. Coupling comes only though the nonlinear part $\hat{a}_{\vec{k}}(t)$, which depends on the whole set of variables $\hat{A}_{\vec{k}}(t)$.

Before proceeding further, let us compare the structure of (11.69) with the situation we had for finite differences. Consider a PDE with linear spatial derivatives and a local nonlinear term (i.e., a nonlinear term that depends on the values of the fields at the same spatial location). In finite differences, the coupling between the coarse-grained variables $A_{\vec{n}}$ comes from the spatial derivatives, so even if the spatial derivatives appear only linearly, the variables are coupled. On the contrary, a local nonlinear term does not lead to coupling. In pseudospectral methods based on Fourier modes, the linear spatial derivatives do not lead to any coupling, while the nonlinear term, even if it is local in real space, couples all the Fourier modes.

11.5.1 Evaluation of the Nonlinear Term

Since typically the nonlinear term couples all the Fourier modes, its evaluation in Fourier space is quite time consuming. Instead, usually, the nonlinear term can be computed easily in real space. Let us consider some typical examples of nonlinear terms and the procedure to evaluate them.

- For a nonlinear term that depends only on the field and not on its spatial derivatives, for example, a nonlinear term of the form $\mathcal{N}[u] = u(x, t)^n$, at each time step one computes the field in real space using (G.17), then evaluates the nonlinear term in real space, and the result is finally transformed to the Fourier space. Thus, $a_k(t)$ will be computed as

  $$a_k = \mathcal{F}_D[(\mathcal{F}_D^{-1}[\hat{u}_k(t)])^n]. \tag{11.70}$$

  The same procedure applies for a transcendental function of the field; for instance, for $\mathcal{N}[u] = \ln(u(x, t))$

  $$a_k = \mathcal{F}_D[\ln(\mathcal{F}_D^{-1}[\hat{u}_k(t)])]. \tag{11.71}$$

- For nonlinear terms that depend on a spatial derivative of the field, such as (11.58), one first evaluates the spatial derivatives in Fourier space. For

example, $\partial^{m+n}A(\vec{x},t)/\partial x_i^n \partial x_j^m$ is evaluated as $(-\mathbf{i}2\pi k_i/L)^n(-\mathbf{i}2\pi k_j/L)^m \hat{A}_{\vec{k}}$. The result is transformed to real space where the nonlinear term is evaluated, and the outcome is finally transformed to the Fourier space. For instance, the nonlinear term for (11.56) is computed as

$$\hat{a}_k(t) = -\mathcal{F}_D\left[\left(\mathcal{F}_D^{-1}[\mathbf{i}q_k\hat{u}_k(t)]\right)^2\right] \tag{11.72}$$

where q_k is given by (11.68).

- A third possibility is that the nonlinear term depends on the field and also on one spatial derivative or several derivatives. At each time step, one first evaluates the spatial derivatives in the Fourier space. Then the field and the spatial derivatives are separately transformed to the real space. The nonlinear term is evaluated in the real space, and the outcome is finally transformed back to the Fourier space. Notice that in this case at each time step it is required to perform several Fourier transformations from the Fourier to the real and then one back from the real space to the Fourier space. For example, for $\mathcal{N}[u] = u(x,t)\partial u(x,t)/\partial x$, $a_k(t)$ will be computed as

$$a_k = \mathcal{F}_D[\mathcal{F}_D^{-1}[\hat{u}_k(t)]\mathcal{F}_D^{-1}[\mathbf{i}q_k\hat{u}_k(t)]] \tag{11.73}$$

which requires two transformations from the Fourier space to the real space.

- If the differential equation includes the spatial derivative of a nonlinear term, (see, e.g., Exercise 8), then one evaluates the nonlinear term in real space, following the guidelines indicated before, transforms the outcome back to the Fourier space, and finally performs the spatial derivative in the Fourier space. For instance, for $\mathcal{N}[u] = \partial u(x,t)^5/\partial x$, $a_k(t)$ will be computed as

$$\hat{a}_k(t) = \mathbf{i}q_k\mathcal{F}_D\left[\left(\mathcal{F}_D^{-1}[\hat{u}_k(t)]\right)^5\right]. \tag{11.74}$$

11.5.2 Storage of the Fourier Modes

In practice, when writing down ω_k or q_k one has to know the order in which the fast Fourier routine returns the Fourier modes, as discussed in Appendix G. In particular, for a one-dimensional real field with $N = 2M$ discretization points, the fast Fourier transform routines typically take advantage of the symmetries of the Fourier transform to return the result in a compact way. Many fast Fourier routines take as input the double precision vector

$$(u_0, u_1, \ldots, u_{N-1}) \tag{11.75}$$

and return a double precision vector of the same length in which the Fourier transform is stored as

$$(\hat{u}_0, \hat{u}_{\frac{N}{2}}, \mathrm{Re}\,(\hat{u}_1), \mathrm{Im}\,(\hat{u}_1), \mathrm{Re}\,(\hat{u}_2), \mathrm{Im}\,(\hat{u}_2), \ldots, \mathrm{Re}\,(\hat{u}_{\frac{N}{2}-1}), \mathrm{Im}\,(\hat{u}_{\frac{N}{2}-1})). \tag{11.76}$$

Note that, for a real input, $\hat{u}_0$ and $\hat{u}_{N/2}$ must be real. Some libraries provide in-place and out-of-place versions of the routines. The former overwrite the original vector with the result, while out-of-place routines return the output in a different

vector. Nevertheless, for a given library, the order of the elements is typically the same.

The vector ω_k should be stored in the same order as the one used by the fast Fourier transform. In the case that the routine stores Fourier modes as (11.76), ω_k should be a double precision vector with the components

$$(\omega_0, \omega_{\frac{N}{2}}, \mathrm{Re}\,(\omega_1), \mathrm{Im}\,(\omega_1), \mathrm{Re}\,(\omega_2), \mathrm{Im}\,(\omega_2), \ldots, \mathrm{Re}\,(\omega_{\frac{N}{2}-1}), \mathrm{Im}\,(\omega_{\frac{N}{2}-1})). \quad (11.77)$$

Similarly, for the vector $\mathbf{i}q_k = 2\mathbf{i}\pi k/L$ required to perform the derivative in Fourier space (note that according to (11.68) $q_{N/2} = 0$)

$$(0, 0, 0, 2\pi/L, 0, 4\pi/L, \ldots, 0, 2\pi(N/2-1)/L). \quad (11.78)$$

11.5.3 Exact Integration of the Linear Terms

Now that we have established a way to evaluate the nonlinear term, the set (11.64) can be numerically solved using any standard method for ordinary differential equations. However, before doing so, let us revise again the structure of (11.64). For PDEs whose linear part include spatial derivatives of high order, the dispersion relation ω_k contains powers of k of high order. To be precise, if the linear operator $\mathcal{L}$ includes spatial derivatives up to order n, then ω_k is a polynomial in k of order n. Thus, for Fourier modes with a large wave number ω_k can take a large value and this typically limits the integration time step, at least in explicit algorithms. These high wave number modes correspond to variations in space at the discretization scale in which we are not interested. In the original partial differential these high wave number modes are damped, but in the numerical evaluation if one uses an explicit method, the integration time step is usually limited by the largest $|\omega_k|$ to avoid numerical instabilities. For instance, in the Nikolaevskii equation, ω_k is a sixth-order polynomial in k and the maximum wave number is given by $k_{max} = N/2$, thus the largest $|\omega_k|$ is of order N^6. Differential equations that have many different time scales are technically referred to as stiff. The difficulty associated with the integration of stiff equations is that, to avoid numerical instabilities, one is forced to take very small time steps and, at the same time, in order to see the evolution at the slower time scale, it is necessary to integrate over long times. For instance, for an explicit Euler method to avoid temporal instabilities, one would have to take h on the order N^{-6}, which is of order $(\Delta x)^6$. A way to avoid this inconvenience is to use implicit methods for the temporal evolution, such as the implicit Euler equation (7.94) discussed in Section 7.4, which has unconditional stability. However, there is a more elegant (and accurate) way to proceed. That is to integrate exactly the linear part of the set (11.64) before applying any numerical method. In this way, the temporal evolution generated by the linear part is evaluated exactly. The nonlinear part of the evolution involves all the modes, but the high wave number modes which have small amplitudes practically do not contribute to it, provided they are damped by the linear part, as they should. As a consequence, the nonlinear part evolves at timescales related to those of the relevant modes whose wave number

does not scale with N and are usually much slower. In Section 7.8, we already discussed the exact integration of the linear part for a single variable. In fact, for a single variable, that is of use only in some specific cases where the nonlinear term indeed evolves much slower than the linear one. Here, the timescale separation appears naturally and therefore the method becomes very useful.

11.5.4 Change of Variables

Disregarding the nonlinear part in (11.64), one obtains a set of uncoupled linear equations that can be readily solved:

$$\hat{u}_k = \mathrm{e}^{\omega_k t}\hat{u}_k(0). \tag{11.79}$$

Based on the form of the solution of the linear equation, we introduce a new variable $z_k(t)$ defined by

$$\hat{z}_k(t) = \mathrm{e}^{-\omega_k t}\hat{u}_k(t). \tag{11.80}$$

Substituting this into the set (11.64), one finds that the new variables $z_k(t)$ satisfy

$$\frac{d\hat{z}_k}{dt} = \mathrm{e}^{-\omega_k t}\hat{a}_k(t). \tag{11.81}$$

To this equation we can apply any standard algorithm for ordinary differential equations.

11.5.5 Heun Method

As a first example, we consider the second-order Heun method which for ordinary differential equations can be written as (7.146). For (11.81), we have

$$\begin{aligned}\hat{z}_k^{(1)} &= \hat{z}_k(t_i) + h\mathrm{e}^{-\omega_k t_i}\hat{a}_k(t_i),\\ \hat{z}_k(t_{i+1}) &= \hat{z}_k(t_i) + \frac{h}{2}\left[\mathrm{e}^{-\omega_k t_i}\hat{a}_k(t_i) + \mathrm{e}^{-\omega_k t_{i+1}}\hat{a}_k(t_{i+1})^{(1)}\right]\end{aligned} \tag{11.82}$$

where, following the notation used in Section 7.8, by $\hat{a}_k(t_{i+1})^{(1)}$ we mean that the nonlinear term $\hat{a}_k(t)$ at time t_{i+1} is evaluated using the set of variables $\hat{z}_k^{(1)}$ with $k = -N/2+1, \ldots, N/2$. Using (11.80), one gets

$$\begin{aligned}\hat{u}_k^{(1)} &= \mathrm{e}^{\omega_k h}\hat{u}_k(t_i) + h\mathrm{e}^{\omega_k h}\hat{a}_k(t_i),\\ \hat{u}_k(t_{i+1}) &= \mathrm{e}^{\omega_k h}\hat{u}_k(t_i) + \frac{h}{2}\left[\mathrm{e}^{\omega_k h}\hat{a}_k(t_i) + \hat{a}_k(t_{i+1})^{(1)}\right].\end{aligned} \tag{11.83}$$

The core of the Heun method with exact integration of the linear part, implementing the evolution on one time step, can be programmed as follows:

```
call nonlinear(t,u,a0,q,N)
u1=w*(u+h*a0)
call nonlinear(t+h,u1,a1,q,N)
```

```
u=w*u+h_half*(w*a0+a1)
t=t+h.
```

In this code, `u` is a real vector of length N that at the beginning of the step contains $\hat{u}_k(t_i)$ and at the end of the step has $\hat{u}_k(t_{i+1})$, which will be used as starting point in the following time step. `u1` is also a real vector that stores $\hat{u}_k(t_{i+1})^{(1)}$. The vector `w` stores $e^{\omega_k h}$. Note that, if the fast Fourier transformation stores the output as (11.76), then ω_k should be stored as (11.77). We use the vectors `a0` and `a1` to store the nonlinear term. `h_half` corresponds to $h/2$.

The subroutine `nonlinear` returns the nonlinear term. It is called twice, once to evaluate the nonlinear function $\hat{a}_k(t_i)$, and once to evaluate $\hat{a}_k(t_{i+1})^{(1)}$. The nonlinear subroutine for the first version of Nikolaevskii equation (see (11.72)) is as follows:

```
subroutine nonlinear(t,u,a,q,N)
implicit none
double precision, dimension (:) :: u,a,q
double precision  :: t
integer :: N
a=q*u
call fft1d(a,N,-1)
a=-a*a
call fft1d(a,N,1)
return
end
```

where `q` stores iq_k in a suitable way, namely, if the fast Fourier transform stores the output as (11.76), then iq_k should be stored as (11.78).

11.5.6 Midpoint Runge–Kutta Method

We now consider implementing the second-order midpoint Runge–Kutta method (7.154). For (11.81), we have

$$\begin{aligned}\hat{z}_k^{(1)} &= \hat{z}_k(t_i) + \frac{h}{2} e^{-\omega_k t_i} \hat{a}_k(t_i) \\ \hat{z}_k(t_{i+1}) &= \hat{z}_k(t_i) + h e^{-\omega_k (t_i + h/2)} \hat{a}_k(t_i + h/2)^{(1)},\end{aligned} \tag{11.84}$$

where, following the notation used in Section 7.8, by $\hat{a}_k(t_i + h/2)^{(1)}$ we mean that the nonlinear term $\hat{a}_k(t)$ at time $t_i + h/2$ is evaluated using the set of variables $\hat{z}_k^{(1)}$ with $k = -N/2 + 1, \ldots, N/2$. Using (11.80), we can write the Runge–Kutta method in terms of the variables $\hat{u}_k$:

$$u_k^{(1)} = e^{\omega_k h/2} \left[\hat{u}_k(t_i) + \frac{h}{2} \hat{a}_k(t_i) \right] \tag{11.85}$$

$$\hat{u}_k(t_{i+1}) = e^{\omega h} \hat{u}_k(t_i) + h e^{\omega_k h/2} \hat{a}_k(t_i + h/2)^{(1)}. \tag{11.86}$$

The core of the method implementing the evolution on one time step can be programmed as follows:

```
call nonlinear(t,u,a0,q,N)
u1=w_half*(u+h_half*a0)
call nonlinear(t+h/2,u1,a0,q,N)
u=w*u+h*w_half*a0
t=t+h
```

where the vector `w_half` stores $e^{\omega_k h/2}$ and the other variables have the same meaning as for the Heun method above. Here, we reuse the vector `a0` to store the nonlinear term at t_i and at $t_i + h/2$. The subroutine `nonlinear` returning the nonlinear term is called twice, once to evaluate the nonlinear function $\hat{a}_k(t_i)$, and once to evaluate $\hat{a}_k(t_i + h/2)^{(1)}$.

11.5.7 Predictor–Corrector

We now implement the fourth-order Adams–Bashford–Moulton predictor–corrector method (7.160) and (7.161) to integrate (11.81). The predictor and corrector algorithms for our equation in the variables $\hat{z}_k$ read, respectively,

$$\hat{z}_k(t_{i+1})^p = \hat{z}_k(t_i) + \frac{h}{24}\left[55e^{-\omega_k t_i}\hat{a}_k(t_i) - 59e^{-\omega_k t_{i-1}}\hat{a}_k(t_{i-1}) + 37e^{-\omega_k t_{i-2}}\hat{a}_k(t_{i-2}) - 9e^{-\omega_k t_{i-3}}\hat{a}_k(t_{i-3})\right] \tag{11.87}$$

and

$$\hat{z}_k(t_{i+1}) = \hat{z}_k(t_i) + \frac{h}{24}\left[9e^{-\omega_k t_{i+1}}\hat{a}_k(t_{i+1})^p + 19e^{-\omega_k t_i}\hat{a}_k(t_i) - 5e^{-\omega_k t_{i-1}}\hat{a}_k(t_{i-1}) + e^{-\omega_k t_{i-2}}\hat{a}_k(t_{i-2})\right]. \tag{11.88}$$

Using (11.80), we can write the integration method in terms of the variables $\hat{u}_k$:

$$\hat{u}_k(t_{i+1})^p = e^{\omega_k h}\left[\hat{u}_k(t_i) + \frac{h}{24}\left(55\hat{a}_k(t_i) - 59e^{\omega_k h}\hat{a}_k(t_{i-1}) + 37e^{2\omega_k h}\hat{a}_k(t_{i-2}) - 9e^{3\omega_k h}\hat{a}_k(t_{i-3})\right)\right] \tag{11.89}$$

and

$$\hat{u}_k(t_{i+1}) = e^{\omega_k h}\hat{u}_k(t_i) + \frac{h}{24}\left[9\hat{a}_k(t_{i+1})^p + 19e^{\omega_k h}\hat{a}_k(t_i) - 5e^{2\omega_k h}\hat{a}_k(t_{i-1}) + e^{3\omega_k h}\hat{a}_k(t_{i-2})\right]. \tag{11.90}$$

The core of the predictor–corrector method implementing the evolution on one time step can be programmed as follows:

```
call nonlinear(t,u,a0,q,N)
v=w*(u+dt55*a0-dt59*a1+dt37*a2-dt9*a3)
call nonlinear(t+h,v,ap,q,N)
u=dt9*ap+w*(u+dt19*a0-dt5*a1+dt1*a2)
a3=a2*w
a2=a1*w
a1=a0*w
t=t+h
```

As in the examples given before, `u` is a real vector of length N, which at the beginning of the step contains $\hat{u}_k(t_i)$ and after the evaluation of the corrector has $\hat{u}_k(t_{i+1})$, which will be used in the following time step. `v` is also a real vector that stores $\hat{u}_k(t_{i+1})^p$. The vector `w` stores $e^{\omega_k h}$. We use the vectors `a0`, `a1`, `a2`, and `a3` to store the nonlinear terms $\hat{a}_k(t_i)$, $e^{\omega_k h}\hat{a}_k(t_{i-1})$, $e^{2\omega_k h}\hat{a}_k(t_{i-2})$, and $e^{3\omega_k h}\hat{a}_k(t_{i-3})$, respectively. At the end of the step, these vectors are multiplied by `w` so that `a2` is converted to `a3`, and so on. The vector `ap` stores $\hat{a}_k(t_{i+1})^p$. The coefficient `dt55` corresponds to $55h/24$, and so on.

The subroutine `nonlinear` returning the nonlinear term is called twice, once to evaluate the nonlinear function $\hat{a}_k(t_i)$ in the predictor equation, and once to evaluate $a_k(t_{i+1})^p$ in the corrector equation.

We recall that multistep methods such as the Adams–Bashford–Moulton have to be warmed up by generating the first time steps using a single step method. In particular, the Adams–Bashford–Moulton requires the knowledge of four time steps before it can start. This can be done using, for example, a fourth-order Runge–Kutta (to be discussed below) or, alternatively, using a lower order method with a much smaller time step.

The Nikolaevskii equation can be written in terms of the variable $v(x,t) = \frac{\partial u(x,t)}{\partial x}$ (see Exercise 8). Figure 11.2 shows a typical space–time configuration for the field $v(x,t)$ in the steady chaotic regime.

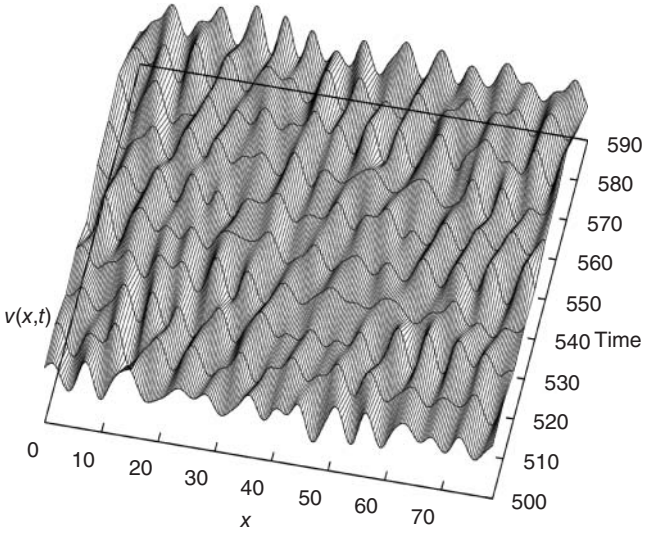


Figure 11.2 Typical space–time configuration for the field $v(x,t)$ fulfilling the Nikolaevskii equation (see Exercise 8) in the steady, chaotic regime. The configuration shown has evolved from a random initial condition within a system size $L = 78$ and $\epsilon = 0.5$. (*Source:* Reprinted from [54].)

11.5.8 Fourth-Order Runge–Kutta

The fourth-order Runge–Kutta method is arguably one of the most widely used methods for the integration of ordinary differential equations. For an equation of the form $\dot{z} = f(t, z)$, it reads

$$\begin{aligned}
\mathcal{K}^{(1)} &= hf(t_i, z(t_i)),\\
\mathcal{K}^{(2)} &= hf\left(t_i + \frac{h}{2}, z(t_i) + \frac{\mathcal{K}^{(1)}}{2}\right),\\
\mathcal{K}^{(3)} &= hf\left(t_i + \frac{h}{2}, z(t_i) + \frac{\mathcal{K}^{(2)}}{2}\right),\\
\mathcal{K}^{(4)} &= hf\left(t_i + h, z(t_i) + \mathcal{K}^{(3)}\right),\\
z(t_{i+1}) &= z(t_i) + \frac{\mathcal{K}^{(1)}}{6} + \frac{\mathcal{K}^{(2)}}{3} + \frac{\mathcal{K}^{(3)}}{3} + \frac{\mathcal{K}^{(4)}}{6} + O[h^5]\,.
\end{aligned} \tag{11.91}$$

For (11.81), one has

$$\begin{aligned}
\mathcal{K}_k^{(1)} &= h\mathrm{e}^{-\omega_k t_i}\hat{a}_k(t_i),\\
\mathcal{K}_k^{(2)} &= h\mathrm{e}^{-\omega_k(t_i+h/2)}\hat{a}_k(t_i + h/2)^{(1)},\\
\mathcal{K}_k^{(3)} &= h\mathrm{e}^{-\omega_k(t_i+h/2)}\hat{a}_k(t_i + h/2)^{(2)},\\
\mathcal{K}_k^{(4)} &= h\mathrm{e}^{-\omega_k(t_i+h)}\hat{a}_k(t_{i+1})^{(3)},\\
\hat{z}_k(t_{i+1}) &= \hat{z}_k(t_i) + \frac{h}{6}\mathcal{K}_k^{(1)} + \frac{h}{3}\mathcal{K}_k^{(2)} + \frac{h}{3}\mathcal{K}_k^{(3)} + \frac{h}{6}\mathcal{K}_k^{(4)}
\end{aligned} \tag{11.92}$$

where $\hat{a}_k(t_i + h/2)^{(1)}$ stands for the nonlinear term $\hat{a}_k(t)$ at time $t_i + h/2$ evaluated using $\hat{z}_k^{(1)} \equiv \hat{z}_k(t_i) + h\mathcal{K}_k^{(1)}/2$ and similarly for $\hat{a}_k(t_i + h/2)^{(2)}$, while $\hat{a}_k(t_{i+1})^{(3)}$ stands for $\hat{a}_k(t)$ at time t_{i+1} evaluated using $\hat{z}_k^{(3)} \equiv \hat{z}_k(t_i) + h\mathcal{K}_k^{(3)}$.

Using (11.80), we can write the Runge–Kutta method in terms of the variables $\hat{u}_k$. The notation can be simplified by introducing

$$\begin{aligned}
\hat{u}_k^{(1)} &= \mathrm{e}^{\omega_k h/2}\left[\hat{u}_k(t_i) + \frac{h}{2}\hat{a}_k(t_i)\right],\\
\hat{u}_k^{(2)} &= \mathrm{e}^{\omega_k h/2}\hat{u}_k(t_i) + \frac{h}{2}\hat{a}_k(t_i + h/2)^{(1)},\\
\hat{u}_k^{(3)} &= \mathrm{e}^{\omega_k h}\hat{u}(t_i) + h\mathrm{e}^{\omega_k h/2}\hat{a}_k(t_i + h/2)^{(2)}
\end{aligned} \tag{11.93}$$

where $\hat{a}_k(t_i + h/2)^{(j)}$ refers to the nonlinear term being evaluated using $\hat{u}_k^{(j)}$. The final algorithm for the u variable is as follows:

$$\begin{aligned}
\hat{u}_k(t_{i+1}) = \mathrm{e}^{\omega h}\left[\hat{u}_k(t_i) + \frac{h}{6}\hat{a}_k(t_i)\right] + \frac{h}{3}\mathrm{e}^{\omega_k h/2}\hat{a}_k(t_i + h/2)^{(1)}\\
+ \frac{h}{3}\mathrm{e}^{\omega_k h/2}\hat{a}_k(t_i + h/2)^{(2)} + \frac{h}{6}\hat{a}_k(t_{i+1})^{(3)}.
\end{aligned} \tag{11.94}$$

The core of the Runge–Kutta method implementing the evolution on one time step can be programmed as follows:

```
call nonlinear(t,u,u4,q,N)
v=w*(u+dt6*u4)
u4=w2*(u+dt2*u4)
call nonlinear(t+dt2,u4,u4,q,N)
v=v+w2*dt3*u4
u4=w2*u+dt2*u4
call nonlinear(t+dt2,u4,u4,q,N)
v=v+w2*dt3*u4
u4=w*u+dt*w2*u4
call nonlinear(t+dt,u4,u4,q,N)
u=v+dt6*u4
t=t+dt
```

where, as usual, `u` is a real vector of length N which at the beginning of the step contains $\hat{u}_k(t_i)$ and at the end has $\hat{u}_k(t_{i+1})$, which will be used in the following time step. `v` is also a real vector that is used to perform the sum of the terms. The vectors `w` and `w2` store $e^{h\omega_k}$ and $e^{h\omega_k/2}$, respectively. The auxiliary vector `u4` is used to store several intermediate operations. The coefficient `dt2` corresponds to $h/2$, and so on. The fourth-order Runge–Kutta calls the subroutine `nonlinear` four times at each time step, double that for the predictor–corrector. Since the evaluation of the nonlinear term is usually computationally costly, the predictor–corrector is, in many instances, preferable.

11.6
Pseudospectral Algorithms for Stochastic Differential Equations

We now consider a field whose dynamics is given by a stochastic PDE of the form

$$\frac{\partial A(\vec{x},t)}{\partial t} = \mathcal{L}[A(\vec{x},t)] + \mathcal{N}[A(\vec{x},t)] + \sqrt{D}\xi(\vec{x},t), \tag{11.95}$$

where, as in Section 11.3, $\xi(x,t)$ is a Gaussian white noise in space and time with zero mean and correlation given by (11.2). For instance, for the KPZ equation (11.3) one has

$$\mathcal{L}[h(\vec{x},t)] = \nu\nabla^2 h(\vec{x},t) \tag{11.96}$$

$$\mathcal{N}[h(\vec{x},t)] = \frac{\lambda}{2}\left|\nabla h(\vec{x},t)\right|^2. \tag{11.97}$$

In particular, in one spatial dimension

$$\mathcal{L}[h(x,t)] = \nu\frac{\partial^2 h(x,t)}{\partial x^2} \tag{11.98}$$

$$\mathcal{N}[h(x,t)] = \frac{\lambda}{2}\left(\frac{\partial h(x,t)}{\partial x}\right)^2. \tag{11.99}$$

Note that the nonlinear term has the same form as that of the Nikolaevskii equation (11.58).

As in Section 11.5, we consider that the field $u(\vec{x}, t)$ satisfies periodic boundary conditions in the interval $x \in [0, L]^d$, and we take the Fourier transform to obtain

$$\frac{\partial \tilde{A}(\vec{q}, t)}{\partial t} = \omega(\vec{q})\tilde{A}(\vec{q}, t) + \tilde{a}(\vec{q}, t) + \tilde{\xi}(\vec{q}, t) \tag{11.100}$$

where the Fourier transform of the noise is defined using the general relation

$$\tilde{\xi}(\vec{q}, t) = \frac{1}{L^d} \int_{[0,L]^d} \mathrm{e}^{\mathrm{i}\vec{q}\,\vec{x}} \xi(\vec{x}, t) d\vec{x}. \tag{11.101}$$

Notice that, being the Fourier transform of a real field, it satisfies $\tilde{\xi}(\vec{q}, t) = \tilde{\xi}^*(-\vec{q}, t)$. One can verify (see Exercise 9) that $\tilde{\xi}(\vec{q}, t)$ are complex Gaussian variables of zero mean, and correlations given by

$$\langle \tilde{\xi}(\vec{q}_{\vec{k}}, t)\tilde{\xi}(\vec{q}_{\vec{k}'}, t')\rangle = \frac{1}{L^d}\delta_{\vec{k}+\vec{k}'}\delta(t - t') \tag{11.102}$$

where, as in Section 11.5, $\vec{q}_{\vec{k}} = 2\pi\vec{k}/L$.

For instance, for the KPZ equation (11.3) in a d-dimensional space, one has

$$\frac{\partial \tilde{h}(\vec{q}, t)}{\partial t} = \omega(\vec{q})\tilde{h}(\vec{q}, t) + \tilde{a}(\vec{q}, t) + \tilde{\xi}(\vec{q}, t) \tag{11.103}$$

where

$$\omega(\vec{q}) = -\nu \sum_{j=1}^{d} q_j^2 \tag{11.104}$$

and

$$\begin{aligned} \tilde{a}(\vec{q}, t) = \mathcal{F}_P\left[\mathcal{N}[h(\vec{x}, t)]\right] &= \mathcal{F}_P\left[\frac{\lambda}{2}\left|\mathcal{F}_P^{-1}[\mathrm{i}\vec{q}\tilde{h}(\vec{q}, t)]\right|^2\right] \\ &= \mathcal{F}_P\left[\frac{\lambda}{2}\sum_{j=1}^{d}\left(\mathcal{F}_P^{-1}[\mathrm{i}q_j\tilde{h}(\vec{q}, t)]\right)^2\right]. \end{aligned} \tag{11.105}$$

As we did in the previous section for deterministic PDEs, we replace the continuum Fourier transform by the discrete one. From (11.100), one has

$$\frac{d\hat{A}_{\vec{k}}(t)}{dt} = \omega_{\vec{k}}\hat{A}_{\vec{k}}(t) + \hat{a}_{\vec{k}}(t) + \hat{\xi}_{\vec{k}}(t) \tag{11.106}$$

where $\hat{\xi}_{\vec{k}}(t)$ is the discrete Fourier transform of the coarse-grained noise $\xi^{cg}_{\vec{n}}(t)$ introduced in Section 11.3, $\hat{\xi}_{\vec{k}}(t) = \mathcal{F}_D[\xi^{cg}_{\vec{n}}(t)]$. The correlations of $\hat{\xi}_{\vec{k}}(t)$ can be computed using its definition

$$\begin{aligned} \langle \hat{\xi}_{\vec{k}}(t)\hat{\xi}_{\vec{k}'}(t')\rangle &= \sum_{\vec{n}}\sum_{\vec{n}'} \mathrm{e}^{\frac{2\pi i}{N}(\vec{n}\vec{k}+\vec{n}'\vec{k}')} \left\langle \xi_{\vec{n}}(t)\xi_{\vec{n}'}(t')\right\rangle \\ &= \sum_{\vec{n}}\sum_{\vec{n}'} \mathrm{e}^{\frac{2\pi i}{N}(\vec{n}\vec{k}+\vec{n}'\vec{k}')} \frac{1}{(\Delta x)^d}\delta_{\vec{n},\vec{n}'}\delta(t - t') \\ &= \frac{1}{(\Delta x)^d}\delta(t - t')\sum_{\vec{n}} \mathrm{e}^{\frac{2\pi i}{N}\vec{n}(\vec{k}+\vec{k}')} = \frac{N^d}{(\Delta x)^d}\delta_{\vec{k}+\vec{k}'}\delta(t - t') \end{aligned} \tag{11.107}$$

where we have used (G.10). We check now that this result corresponds to the continuum one, that is, (11.102), if we make the correspondence

$$\tilde{\xi}(q_k, t) \to \frac{1}{N^d}\hat{\xi}_k(t). \tag{11.108}$$

Indeed

$$\langle \tilde{\xi}(\vec{q}_{\vec{k}}, t)\tilde{\xi}(\vec{q}_{\vec{k}'}, t')\rangle = \left\langle \frac{1}{N^d}\hat{\xi}_{\vec{k}}(t)\frac{1}{N^d}\hat{\xi}_{\vec{k}'}(t')\right\rangle = \frac{1}{(N\Delta x)^d}\delta_{\vec{k}+\vec{k}'}\delta(t-t'), \tag{11.109}$$

which is the expected result (see (11.102)).

In what follows, we will consider a one-dimensional field described by the variable $u(x, t)$ so that one can clearly see the parallelism with what we did in Section 11.5 with the deterministic PDEs. We use the notation given by (11.65), (11.66), and (11.67).

$$\frac{d\hat{u}_k(t)}{dt} = \omega_k\hat{u}_k(t) + \hat{a}_k(t) + \hat{\xi}_k(t) \tag{11.110}$$

For instance, for the KPZ in one spatial dimension (11.4), one has

$$\omega_k = -\left(\frac{2\pi k}{L}\right)^2 \tag{11.111}$$

$$\hat{a}_k(t) = \mathcal{F}_D\left[\mathcal{N}[h(x,t)]\right] = \mathcal{F}_D\left[\frac{\lambda}{2}\left(\mathcal{F}_D^{-1}\left[\mathbf{i}\frac{2\pi k}{L}\hat{h}_k(t)\right]\right)^2\right]. \tag{11.112}$$

Before using any numerical algorithm to solve the set (11.110), we make a change of variables based on the solution of the linear equation, which is an extension of what we did in Section 7.8 for a single variable:

$$\hat{u}_k(t) = \mathrm{e}^{\omega_k t}\hat{z}_k(t) + \mathrm{e}^{\omega_k t}\int_0^t ds\, \mathrm{e}^{-\omega_k s}\hat{\xi}_k(s) \equiv \mathrm{e}^{\omega_k t}\hat{z}_k(t) + \hat{G}_k(t) \tag{11.113}$$

where $\hat{G}_k(t)$ is defined as

$$\hat{G}_k(t) = \mathrm{e}^{\omega_k t}\int_0^t ds\, \mathrm{e}^{-\omega_k s}\hat{\xi}_k(s). \tag{11.114}$$

Then, $\hat{z}_k(t)$ formally satisfies the equation

$$\frac{\partial\hat{z}_k(t)}{\partial t} = \mathrm{e}^{-\omega_k t}\hat{a}_k(t) \tag{11.115}$$

which is formally the same as (11.81) for a deterministic PDE. As discussed in Section 7.8, the noise dependence is still present since the nonlinear term $\hat{a}_k(t)$ depends on $\hat{\xi}_k(t)$ through the relation between $\hat{x}_k(t)$ and $\hat{z}_k(t)$ (11.113). The advantage is, as in the case of a single variable, that the dependence between $\hat{x}_k(t)$ and $\hat{z}_k(t)$ involves an integral of the noise which is better behaved than the noise itself. Therefore we may formally consider that the effect of the noise on the variable $z(t)$ is smooth in some sense and apply to (11.115) the numerical methods as we used for (11.81). In terms of $\hat{z}_k$, the difference equations that one obtains are exactly the same as those for deterministic PDEs in Section 11.5. The difference appears when going from $\hat{z}_k$ back to the $\hat{u}_k$. Now, the change of variables involves

$\hat{G}_k(t)$ as given by (11.113), similar to what we found in Section 7.8 when discussing the numerical methods for (7.145).

11.6.1 Heun Method

As before, we start considering the second-order Heun method which in terms of $\hat{z}_k(t)$ is given by (11.82). Replacing $\hat{z}_k(t) = \mathrm{e}^{-\omega_k t}\left(\hat{u}_k(t) - \hat{G}_k(t)\right)$, one has

$$\begin{aligned}\hat{u}_k^{(1)} &= \mathrm{e}^{\omega_k h}\hat{u}_k(t_i) + h\mathrm{e}^{\omega_k h}\hat{a}_k(t_i) + \hat{g}_k(t_i),\\ \hat{u}_k(t_{i+1}) &= \mathrm{e}^{\omega_k h}\hat{u}_k(t_i) + \frac{h}{2}\left[\mathrm{e}^{\omega_k h}\hat{a}_k(t_i) + \hat{a}_k(t_{i+1})^{(1)}\right] + \hat{g}_k(t_i)\end{aligned} \tag{11.116}$$

where

$$\hat{g}_k(t) \equiv \hat{G}_k(t_{i+1}) - \mathrm{e}^{\omega_k h}\hat{G}_k(t_i) = \mathrm{e}^{\omega_k t_{i+1}}\int_{t_i}^{t_{i+1}} \mathrm{e}^{-\omega_k s}\hat{\xi}_k(s)ds \tag{11.117}$$

which is a generalization to several variables of the stochastic process $g(t)$ defined in Section 7.8 for a single variable (see (7.150)). The processes $g_k(t)$ are Gaussian with zero mean and correlation

$$\left\langle \hat{g}_k(t_i)\hat{g}_{k'}(t_j)\right\rangle = \frac{\mathrm{e}^{2\omega_k h}-1}{2\omega_k}\frac{N}{\Delta x}\delta_{k+k'}\delta_{ij} \tag{11.118}$$

(we have assumed that $\omega_k = \omega_{-k}$).

In practice, the variables $\hat{g}_k(t)$ can be obtained by writing them as

$$\hat{g}_k(t_i) = \sqrt{\frac{\mathrm{e}^{2\omega_k h}-1}{2\omega_k}\frac{1}{\Delta x}}\,\hat{v}_k(t_i) \tag{11.119}$$

where

$$\hat{v}_k(t_i) = \mathcal{F}_D[v_n(t_i)] \tag{11.120}$$

and $v_n(t_i)$ is a set of independent Gaussian random numbers of zero mean and variance 1. The generation of $\hat{v}_k(t_i)$ can be coded as follows:

```
subroutine g_kvec(xi)
 implicit none
 double precision (:) :: xi
 integer :: i,N
 double precision ran_g
 N=size(xi)
 do i=1,N
   xi(i)=ran_g()
 enddo
 call fft1d(xi,N,1)
end subroutine g_kvec
```

where the vector `xi` should be dimensioned in the program calling this subroutine to the number of discretization cells N. The routine first calls for the generation of N uncorrelated random numbers, which are then Fourier transformed to obtain

N numbers with correlation δ_{k+k}. It is also possible to generate $\hat{\xi}_k$ directly in the Fourier space (see Further Reading and References).

As an example, let us apply the stochastic pseudospectral Heun method to the KPZ equation in one spatial dimension. The subroutine providing the nonlinear part can be programmed as follows:

```
subroutine nonlinear(t,u,a,q,N)
 implicit none
 double precision, dimension (:) :: u,a,q
 double precision  :: t,lambda_half=0.1d0
 integer :: N
 a=q*u
 call fft1d(a,N,-1)
 a=lambda_half*a*a
 call fft1d(a,N,1)
end subroutine nonlinear
```

where `lambda_half` stores $\lambda/2$. Here we implemented this value as a data in the subroutine. Alternatively, one could pass it to the subroutines as one additional variable in the call.

The core of the stochastic pseudospectral Heun method, implementing the evolution on one time step, can be written as follows:

```
call g_kvec(xi)
xi=noise_scale*xi
call nonlinear(t,u,a0,q,N)
u1=xi+w*(u+h*a0)
call nonlinear(t+h,u1,a1,q,N)
u=xi+w*u+h_half*(w*a0+a1)
t=t+h
```

where we call the routine `g_kvec` to generate the Gaussian random numbers in the Fourier space and the vector `noise_scale` stores the strength of the stochastic terms $\sqrt{\frac{e^{2\omega_k h}-1}{2\omega_k \Delta x}}$. The other variables are as in the deterministic Heun method described in Section 11.5.5. The subroutine ''nonlinear'' returning the nonlinear terms in Fourier space is called twice.

11.6.2 Predictor–Corrector

The Adams–Bashford–Moulton predictor–corrector method is given in terms of $\hat{z}_k(t)$ by (11.87) and (11.88). Using (11.113), one has

$$\hat{u}_k(t_{i+1})^p = \hat{g}_k(t_i) + e^{\omega_k h}\left[\hat{u}_k(t_i) + \frac{h}{24}\left(55\hat{a}_k(t_i) - 59e^{\omega_k h}\hat{a}_k(t_{i-1}) + 37e^{2\omega_k h}\hat{a}_k(t_{i-2}) - 9e^{3\omega_k h}\hat{a}_k(t_{i-3})\right)\right] \tag{11.121}$$

and

$$\hat{u}_k(t_{i+1}) = \hat{g}_k(t_i) + e^{\omega_k h}\hat{u}_k(t_i) + \frac{h}{24}\left[9\hat{a}_k(t_{i+1})^p + 19e^{\omega_k h}\hat{a}_k(t_i) - 5e^{2\omega_k h}\hat{a}_k(t_{i-1}) + e^{3\omega_k h}\hat{a}_k(t_{i-2})\right]. \quad (11.122)$$

The stochastic predictor–corrector can be implemented by just adding at each time step the stochastic terms to the deterministic evolution. program. The core is as follows:

```
call g_kvec(xi)
xi=noise_scale*xi
call nonlinear(t,u,a0,q,N)
v=xi+w*(u+dt55*a0-dt59*a1+dt37*a2-dt9*a3)
call nonlinear(t+h,v,ap,q,N)
u=xi+dt9*ap+w*(u+dt19*a0-dt5*a1+dt1*a2)
a3=a2*w
a2=a1*w
a1=a0*w
t=t+h
```

where `g\_kvec`, xi and `noise\_scale` are as described in 11.6.1 and the other variables are as in the deterministic pseudospectral predictor-corrector method described in 11.5.7.

11.7 Errors in the Pseudospectral Methods

It is important to analyze the sources of errors in pseudospectral methods. Even when the time integrator (Runge–Kutta, predictor–corrector, or other) is stable, there is always the question of whether the solution of the pseudospectral method converges towards the (exact) solution of the system. The most important problem is aliasing, which is discussed in Appendix G. It is always a good practice to plot the power spectra of the field to check that large wave number Fourier modes have a very small amplitude. The second largest source of error, which is in fact related to aliasing, is that, even if the approximant at $t = 0$ is a very good one, it might get bad as time goes on. This problem already appears in linear equations and one spatial dimension. Consider

$$\frac{\partial u(x,t)}{\partial t} = \mathcal{L}[u] \quad (11.123)$$

with the initial condition $u(x,0) = u^0(x)$ and periodic boundary conditions in the interval $(0, L)$. This equation, in the Fourier space, adopts the form

$$\frac{\partial \tilde{u}(q,t)}{\partial t} = \omega(q)\tilde{u} \quad (11.124)$$

whose solution is

$$\tilde{u}(q,t) = \tilde{u}^0(q)\mathrm{e}^{\omega(q)t}. \tag{11.125}$$

The solution in real space is obtained by using the inverse transform in the series form:

$$u(x,t) = \sum_{k=-\infty}^{\infty} \tilde{u}_k^0 \mathrm{e}^{-\mathrm{i}q_k x}\mathrm{e}^{\omega_k t} \tag{11.126}$$

(recall the notation $F_k = F(q_k)$ for any function $F(q)$). $\tilde{u}_k^0$ is obtained as the Fourier transform of the (periodic) function $u^0(x)$:

$$\tilde{u}_k^0 = \frac{1}{L}\int_{[0,L]} u^0(x)\mathrm{e}^{\mathrm{i}q_k x}dx. \tag{11.127}$$

This solution can be written in a more compact form using a propagator

$$u(x,t) = \int_{[0,L]} dy\, u^0(y)G(x-y,t) \tag{11.128}$$

where

$$G(z,t) = \frac{1}{L}\sum_{k=-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}q_k z+\omega_k t}. \tag{11.129}$$

Using

$$\sum_{k=-\infty}^{\infty} \mathrm{e}^{\mathrm{i}ka} = 2\pi\delta(a) \tag{11.130}$$

it is easy to prove that

$$G(z,t=0) = \delta(z). \tag{11.131}$$

So far, everything is exact. The approximation of replacing the sum in (11.126) by a finite sum is the method of Galerkin:

$$u(x,t) = \sum_{k=-N/2}^{N/2} \tilde{u}_k^0 \mathrm{e}^{-\mathrm{i}q_k x}\mathrm{e}^{\omega_k t}. \tag{11.132}$$

However, this method requires the knowledge of the exact Fourier coefficients $\tilde{u}_k^0$ (or a good numerical evaluation of them).

In the pseudospectral methods one replaces the above expressions by the corresponding ones using the discrete Fourier transform:

$$u(x,t) = \frac{1}{N}\sum_{k=-N/2}^{N/2}{}' \hat{u}_k^0 \mathrm{e}^{-\mathrm{i}q_k x}\mathrm{e}^{\omega_k t} \tag{11.133}$$

where, as in Appendix G, the prime indicates halving of the first and last terms in the sum. $\hat{u}_k^0$ are obtained as the discrete Fourier transform of the initial condition $\hat{u}_k^0 = \mathcal{F}_D[u_n^0]$.

We ask now what is the difference between the exact solution (11.126) and the one given by (11.133). In fact, there is an exact relation between $\tilde{f}_k$ and $\hat{f}_k$:

$$\hat{f}_k = N^d \sum_{m=-\infty}^{\infty} \tilde{f}_{k+mN}. \tag{11.134}$$

Therefore, the numerical solution can be written as

$$u(x,t) = \sum_{k=-N/2}^{N/2}{}' \sum_{m=-\infty}^{\infty} \tilde{u}^0_{k+mN} \mathrm{e}^{-\mathrm{i}q_k x} \mathrm{e}^{\omega_k t}. \tag{11.135}$$

Defining an index j as $j = k + mN$, the double sum is equivalent to a sum over the index j going from $-\infty$ to ∞. The indexes m and k are related to j as $k = j \mod (N)$, and k is the integer part of j/N. Therefore, one can write (11.135) as

$$u(x_n,t) = \sum_{j=-\infty}^{\infty} \tilde{u}^0_j \mathrm{e}^{-\mathrm{i}q_j x_n} E_j(t) \tag{11.136}$$

with

$$E_j(t) = \begin{cases} \mathrm{e}^{\omega_k t} & \text{if } j = k \mod (N), k \in (-N/2+1, N/2-1) \\ \frac{1}{2}\mathrm{e}^{\omega_{N/2} t} + \frac{1}{2}\mathrm{e}^{\omega_{-N/2} t} & \text{for } j = N/2 \mod (N). \end{cases} \tag{11.137}$$

Equation (11.136) is similar to the exact solution (11.126) with the exception that the E_j's are equal to the alias values of $\mathrm{e}^{\omega_j t}$. This means that the modes with wave number $|j| > N/2$ evolve not with the factor ω_j but with the factor of their aliases. For instance, if $N = 8$, the modes $k = -17, -9, -1, 7, 15, 23, 31, \ldots$ all evolve using the factor ω_{-1}. This is a source of errors. If the higher order coefficients $\tilde{u}^0_j$ are very small, then their contribution to the solution is small. If, on the other hand, the function $u^0(x)$ is not smooth (as, e.g., the case of noise white in space), then the convergence toward the exact solution might need a prohibitively large value of N.

In other words, the exact solution (11.126) involves the evolution of infinite Fourier modes. In the pseudospectral method, indeed, infinite modes do appear in the evolution equation, see (11.136), but for modes with wave numbers larger than $N/2$, the amplifying factor is not the correct one. In the Galerkin method, (11.132), the modes with $|k| > N/2$ simply do not appear.

To illustrate this phenomenon, we consider the equation

$$\partial_t u(x,t) = u + \partial_x u \tag{11.138}$$

for which the amplifying factor is complex:

$$\omega(q) = 1 - \mathbf{i}q. \tag{11.139}$$

It is easy to check that the solution, for an arbitrary initial condition, is

$$u(x,t) = \mathrm{e}^t u^0(x+t). \tag{11.140}$$

In the following, we compare the exact solution (11.140) taking as initial condition

$$u(x, t=0) = 3/(5 - 4\cos(10\pi x/L) - 0.25\cos(20\pi x/L)) \tag{11.141}$$

with the pseudospectral numerical solution.

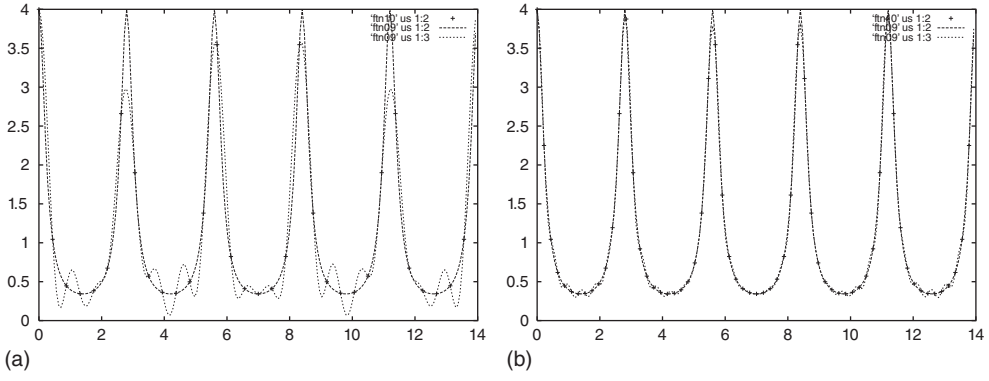


Figure 11.3 Initial condition (11.141) with $L = 14$. The solid line is the exact plot of the initial condition, while symbols represent the value of the exact solution at the discretization points $N = 32$ (a) and $N = 64$ (b). The dashed line shows the representation in real space of the initial condition using only the Fourier modes included in the discretization.

We consider a system of size $L = 14$. Figure 11.3 shows the exact initial condition, the value of the initial condition at the lattice points, and the approximated initial condition obtained using only the Fourier modes included in the discretization. We considered two cases: $N = 32$ (panel (a)), and $N = 64$ (panel (b)). Of course, the approximated initial condition always matches exactly the solution at the lattice points. For $N = 32$, already at $t = 0$ we do not have a very accurate approximation for the initial condition. Accuracy increases when considering $N = 64$ discretization points.

We now let the system evolve up to time $t = 1$. Figure 11.4 shows the results for $N = 32$ lattice points. Now, as a consequence of the dynamical evolution, the numerical solution differs from the exact one even at the discretization points, as can be clearly seen in the lower panel. The situation improves by considering 64 lattice points as shown in Figure 11.5. Now, after $t = 1$, the numerical solution still agrees very well with the exact one. Finally, in Figure 11.6, we show the initial condition and the field after $t = 1$ for $N = 128$. In this case, you can see that the numerical solution provides an excellent representation of the exact solution.

The message is that it is necessary to have a very good resolution for the Fourier modes. So the power spectra of the field should indicate that high wave number modes have very small amplitudes. Still, even if the resolution is fine enough for the initial condition, that does not mean that it will continue to be so as time evolves, since the large modes (which are necessarily incorrect) might increase exponentially with time. Therefore, in practice it is advisable to check from time to time the profile of the power spectra of the field to make sure that as time evolves high wave number modes remain at very low amplitude. In fact, this test can be easily performed in pseudospectral methods, since the dynamical evolution is already calculated in Fourier space. The only

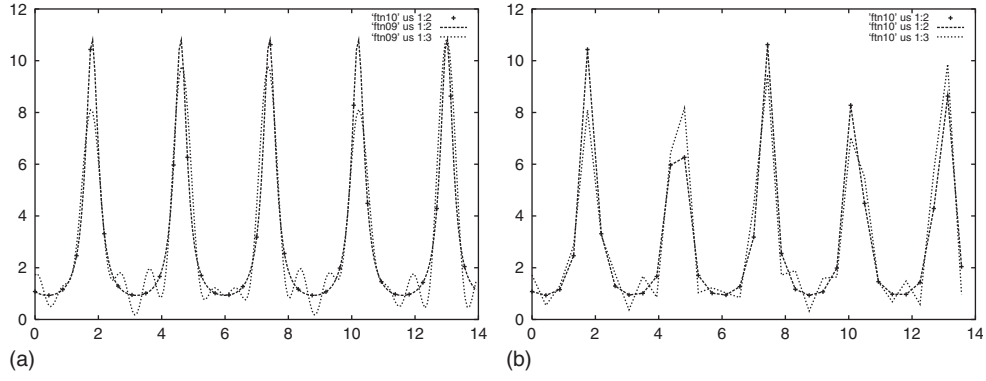


Figure 11.4 Evolution of the field given by (11.138) after time $t = 1$. In panel (a), the solid line shows the exact solution while the dashed line shows the representation in real space of the numerical solution using the Fourier modes included in the discretization with $N = 32$. In panel (b), we have plotted the values at the lattice points and joined them with lines to better appreciate the differences between the exact solution and the numerical result at the lattice points.

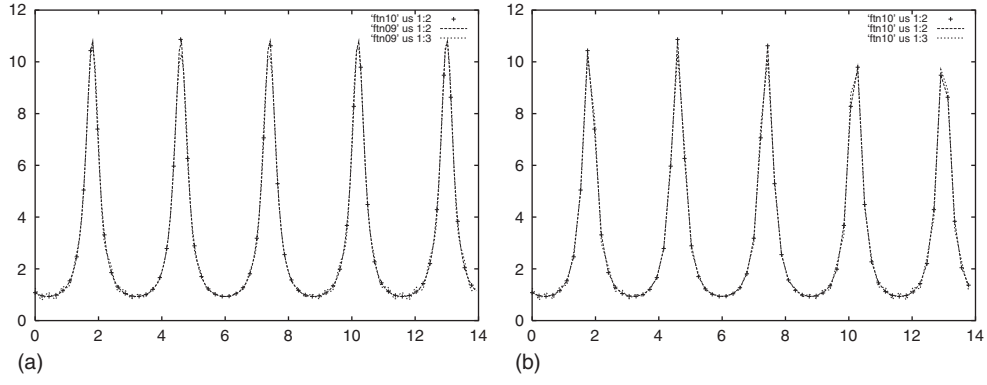


Figure 11.5 As Figure 11.4 but with $N = 64$.

thing that has to be done is to plot from time to time the intensity of the field in the Fourier space, $|\hat{u}_k(t)|^2$. Recall that $\hat{u}_k(i)$ is stored in the vector u in the core programs indicated in Sections 11.5 and 11.6. Particular care has to be taken in systems where localized structures, fronts, or defects are formed, since these structures may have a small size in real space, meaning large wave numbers in Fourier space. The spatial discretization step should always be much smaller than the typical size of these structures, so that the typical wave numbers involved in the description of these structures are, and remain, much smaller than $q_{N/2}$.

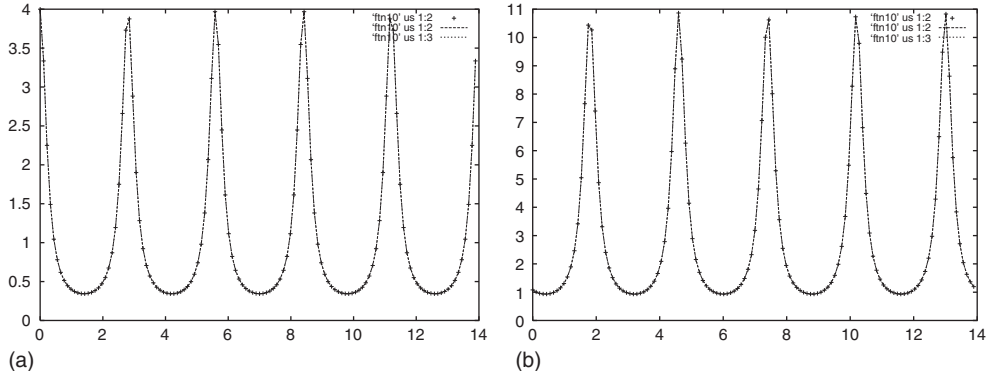


Figure 11.6 Initial condition (11.141) with $L = 14$ (a) and values of the field at the discretization points after $t = 1$ (b) with $N = 128$. The symbols, joined by a solid line, show the exact solution evaluated at the lattice points. The dashed line corresponds to the values at the lattice points of the numerical solution.

Further Reading and References

A general introduction to finite difference methods for PDEs can be found at [5].

The methodology behind the analysis of the numerical stability was first suggested by J. Crank and P. Nicolson in the article where they introduced their integration method (see Exercise 4) [55], reprinted in [56]. In 1950, J. Charney, J.G. Fjørtorft, and J. von Neumann fully developed the stability analysis in order to determine the causes of forecast errors when integrating the barotropic vorticity equation [57].

A broad overview of the effect of fluctuations in spatially extended systems can be found at the book by J. García-Ojalvo and J.M. Sancho [58]. In Appendix B, you can find a methodology to generate Gaussian white noises with correlations $\left\langle \hat{\xi}_k(t_i)\hat{\xi}_{k'}(t_j) \right\rangle = \delta_{k+k'}\delta_{ij}$ directly in the Fourier space.

For a discussion on pseudospectral Fourier methods for deterministic PDEs, see the review [59].

The Nikolaevskii equation was introduced in [60] to describe the propagation of longitudinal seismic waves in viscoelastic media. It has been found that certain class of reaction–diffusion systems can exhibit chemical turbulence equivalent to the one described by the Nikolaevskii equation [61]. Extensive chaos in the Nikolaevskii equation was analyzed in [54].

The Kardar–Parisi–Zhang equation was introduced in [62].

Exercises

1) Consider the upstream derivative method for the advection equation (11.40).
   a. Write down the finite differences equation considering that $v > 0$.

b. Apply the von Neumann ansatz (11.24) to show that

$$\lambda(k) = 1 - \frac{\nu h}{\Delta x}\left[1 - \cos(k\Delta x)\right] - i\sin(k\Delta x).$$

c. Evaluate $|\lambda(k)|^2$ and show that $|\lambda(k)|^2 \leq 1$ if the Courant criterion $\nu h/\Delta x \leq 1$ is fulfilled.

d. Show that for $h = \Delta x/\nu$, perturbations neither grow nor decay, that is, $|\lambda(k)| = 1$ for all k.

e. Assuming $h < \Delta x/\nu$, for which wave number perturbations decay faster?

f. Expand $|\lambda(k)|^2$ for small k and discuss the decay rate of small wave number perturbations.

2) Consider the Heun method to integrate the diffusion equation.

a. Apply the von Neumann ansatz to (11.48) to obtain (11.49).

b. Write (11.49) as a function of $\gamma = Dk/(\Delta x)^2 \sin^2(k\Delta x/2)$ and show that $|\lambda(k)| < 1$ for $0 \leq \gamma < 0.5$.

c. Show that the condition for stability is given by the Courant–Friedrichs–Lewy criterion (11.34).

3) Apply the von Neumann stability analysis to the deterministic part of the difference equation (11.50) corresponding to the Heun method applied to the KPZ equation. Show that the Courant–Friedrichs–Lewy criterion (11.34) is a necessary condition for numerical stability.

4) John Crank and Phyllis Nicolson published in 1947 ''A practical method for numerical evaluation of solutions of partial differential equations of the heat-conduction type.'' The idea behind their method was to discretize the space using centered derivatives and the time using the semi-implicit Euler method, as in (7.95), which is of second order in time. In this way, both spatial and temporal derivatives are evaluated at second order.

a. Show that the diffusion equation (11.28) is discretized as

$$\begin{aligned} A(x_n, t_{i+1}) = -A(x_n, t_i) + d\,\big[&A(x_{n+1}, t_i) - 2A(x_n, t_i) + A(x_{n-1}, t_i) \\ &+ A(x_{n+1}, t_{i+1}) - 2A(x_n, t_{i+1}) + A(x_{n-1}, t_{i+1})\big] \end{aligned}$$

where $d = Dh/2(\Delta x)^2$.

b. Show that the previous expression can be rewritten as

$$\begin{aligned} -\frac{d}{2}A(x_{n+1}, t_{i+1}) + (1+d)A(x_n, t_{i+1}) - \frac{d}{2}A(x_{n-1}, t_{i+1}) \\ = \frac{d}{2}A(x_{n+1}, t_i) + (1-d)A(x_n, t_i) + \frac{d}{2}A(x_{n-1}, t_i) \end{aligned}$$

and discuss how this could be implemented in a computer program.

c. Perform the von Neumann stability analysis to show that

$$\lambda(k) = \frac{1 - 4d[\sin(k\Delta x/2)]}{1 + 4d[\sin(k\Delta x/2)]}$$

and discuss the conditions for stability.

5) Write a program to integrate the KPZ equation with periodic boundary conditions using the Heun algorithm.

a. Consider $L = 10$, $N = 256$, $\nu = 1$, $\lambda = 0$, and $D = 0$. Change the time step and check the stability limit predicted by the Courant criterion.
b. Repeat the previous analysis for $N = 512$. Discuss the results.
c. Consider $\lambda = 0.1$. Starting from a random initial condition, plot the interface at different times.
d. Repeat the previous item for $\lambda = 0.2, 0.5, 1$, and discuss the results.
e. Set $L = 10$, $N = 256$, $\nu = 1$, and $\lambda = 1$, and integrate the equation for $D = 0.01$ and for $D = 0.1$. Compare the shape of the fronts with those obtained in the case $D = 0$.

6) This exercise is to get acquainted with the use of fast Fourier transform routines. It is supposed that you have access to a library in which the fast Fourier Transform is implemented. If you do not have that library, you can, for instance, download and install the public domain FFTW (http://www.fftw.org/).
a. Consider the periodic function $\cos(ax)$. Sample the function with a suitable Δx over a length multiple of the spatial period. Call the direct fast Fourier transform. Discuss how the fast Fourier transform routine orders the output, and plot the power spectra. Compare with the analytic power spectra. Try different discretization sizes.
b. Using the same function, call the direct fast Fourier transform and its inverse. Compare the results with the initial function and discuss the precision of the routine. Try other periodic functions.
c. Consider, again, the function $\cos(ax)$. Call the direct fast Fourier transform and perform the derivative in the Fourier space using (G.24) and (G.25) (see Appendix G). Call the inverse Fourier transform and plot the results in real space. Plot also the analytical result and compare. Try different discretization sizes.
d. Perform the derivative of the function using centered space finite differences and plot the result. Compare with the result obtained in the previous case with the same discretization size.
e. Repeat the previous two items with the second derivative. Discuss the results.

7) Consider the Nikolaevskii equation (11.56) with a random initial condition.
a. Write a computer program to integrate the equation using the pseudospectral Heun method. Integrate the equation for $\epsilon = 0.5$ in a system of size $L = 78$. Plot the spatial profile of the solution at several times. Plot also the power spectrum of the solution at these times. Try different spatial discretization sizes Δx and time steps Δt and choose a suitable ones for which large wave number modes are always small. Discuss the results.
b. Write a computer program to integrate the equation using the pseudospectral Adams–Bashford–Moulton predictor–corrector method. Use the Heun method with a suitable time step (determined above) to warm the predictor corrector. Integrate the equation for $\epsilon = 0.5$ in a system of size $L = 78$ using the same random initial condition as in (a) and compare the results. Discuss the efficiency of the different methods.

c. Explore the effect of ϵ in the dynamics. Try different values for ϵ and plot the spatial profile of the solution and the power spectrum at several times. Verify that the large wave number modes remain damped. Discuss the results.

8) The Nikolaevskii equation is sometimes written in terms of the variable $v(x,t) = \frac{\partial u(x,t)}{\partial x}$.
   a. Show that in terms of the new variable the equation reads
   $$\frac{\partial v(x,t)}{\partial t} = -\frac{\partial^2}{\partial x^2}\left[\epsilon - \left(1 + \frac{\partial^2}{\partial x^2}\right)^2\right] v(x,t) - \frac{\partial v(x,t)^2}{\partial x}\,.$$
   b. Decompose the equation into linear and nonlinear parts. Compare with (11.57) and (11.58).
   c. Write a program to integrate this equation using the pseudospectral Adams–Bashford–Moulton predictor–corrector method warmed with a fourth-order Runge–Kutta.
   d. Integrate numerically the equation with $\epsilon = 0.5$ and $L = 78$. By plotting the spatial profile of the solution and the power spectra for different times, discuss the suitable step size for the spatial and temporal discretization. Compare the results with Figure 11.2.
   e. Explore the effect of ϵ in the dynamics. Try different values for ϵ and plot the spatial profile of the solution and the power spectrum at several times. Verify that the large wave number modes remain damped. Discuss the results and compare with the results obtained in Exercise 7.

9) Consider a spatiotemporal white noise which in the Fourier space is given by (11.102). Consider that we have only one spatial dimension.
   a. Show that $\langle \tilde{\xi}(q,t)\rangle = 0$.
   b. Use
   $$\frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}(q-q_0)x} dq = \delta(q - q_0)$$
   to evaluate the correlation $\langle \tilde{\xi}(q,t)\tilde{\xi}(q',t')\rangle$.
   c. Obtain (11.102).

10) Write a program to integrate the KPZ equation in one spatial dimension with periodic boundary conditions using the pseudospectral stochastic Heun algorithm.
   a. Consider $L = 10$, $N = 256$, $\nu = 1$, $\lambda = 0$, and $D = 0$. Integrate the equation with a random initial condition. Plot the field profile and the power spectra at different times. Check that the high wave number modes are damped. Discuss what happens if the time step is increased.
   b. Consider $\lambda = 0.2$, so that now the system is no longer linear. Consider different values for N and time steps. Plot the field profile and the power spectra. Discuss the results.
   c. Repeat the previous item for $\lambda = 0.5, 1$, and discuss the results. Compare the results and the performance with that of Exercise 5.
   d. Set $L = 10$, $\nu = 1$, $\lambda = 1$, and integrate the equation for $D = 0.01$ and for $D = 0.1$. Plot the shape of the fronts and compare with Exercise 5.

11) For the KPZ equation, one can define a mean value $\overline{h}(t, L)$ and a width $W(t, L)$ for the interface as

$$\overline{h}(t, L) = \langle h(x, t) \rangle \,, \qquad W(t, L) = \sqrt{\left\langle \left(h(x, t) - \langle h(x, t) \rangle \right)^2 \right\rangle},$$

where $\langle ... \rangle$ stands for averages over space and over noise realizations.

a. Write a program to integrate the KPZ equation in one spatial dimension with periodic boundary conditions using the pseudospectral Adams–Bashford–Moulton predictor–corrector method warmed with the Heun method considered in Exercise 10. Try different time steps. Check everything is fine by comparing with the results of program you used in Exercise 10 and also make sure the high wave number modes remain damped.

b. Consider $L = 10$, $\nu = 1$, $\lambda = 1$, and $D = 0.1$, and evaluate $\overline{h}(t, L)$ and $W(t, L)$ for different times averaging over 50, 100, and 200 noise realizations. Discuss the results.

c. Repeat the previous item for $L = 20$ and $L = 40$. Discuss the results.

Appendix A
Generation of Uniform $\hat{\mathbf{U}}(0,1)$ Random Numbers

A.1
Pseudorandom Numbers

In this appendix, we explain how it is possible to generate $\hat{\mathbf{U}}(0,1)$ independent random numbers, that is, random numbers uniformly distributed in the $(0,1)$ interval that can be efficiently used in any stochastic algorithm, Monte Carlo or Langevin. We cannot, and will not, cover all the vast bibliography in this topic. Our aim is to introduce the reader to the major problems he or she will encounter for the generation of uniform random numbers and the basic families of algorithms that have been proposed, so that the routines that will be used do not appear as ''magic black boxes'' which provide random numbers in an unknown and uncontrollable way.

The first thing we need to realize is that the algorithms for the generation of random numbers are implemented in a computer where real numbers are not stored with infinite precision but the number of decimal places is finite and set from the beginning. Therefore, most random number generators provide them in the form of a fraction $u = m/M$, where m is an integer number uniformly distributed in the interval $[0, M-1]$.[1)] The greater the value of M, the better the approximation of the generated numbers to the true distribution $\hat{\mathbf{U}}(0,1)$.

Once a large value of M has been chosen, the question to answer is: how can we generate integer numbers uniformly distributed in the interval $[0, M-1]$? A legitimate way could be the following: make M identical balls numbered 0 to $M-1$ and place them in a bag. Shake the bag, pick one of the balls, and read the number, say m_0, on the ball. Replace the ball in the bag, shake the bag again, pick one of the balls and read the number, say m_1, on the ball. Replace the ball, shake the bag again . . . , and repeat as many times as random numbers you need.

First of all, it is not clear whether this procedure would produce a set of truly uniform random numbers. Some of the balls might be slightly heavier than the others and have a different probability of being chosen, once replaced. If the shaking is not good enough, the ball might stay near the top of the bag and have a larger

1) In some cases, and in order to avoid some singularities that can appear when computing functions such as the logarithm, it is better to write the uniform random number as $(m + 0.5)/M$. However, this is not very important in most applications.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

chance to be chosen again, and so on. We could certainly improve the procedure. For instance, chose $M = 2^b$, a power of 2, and work in base 2 (a base loved by computers). Then we could manage simply with two balls labeled 0 and 1. For the generation of each number m_i, we would extract b balls and construct that number bit by bit. In any event, even if the procedure were absolutely fair, it is clear that it is not efficient. Recall that we might need a large number (of the order of millions or billions) of random numbers for our numerical work and it would be completely impracticable to obtain them this way. It is best to design some numerical algorithm to generate the values $m_0, m_1, m_2, \ldots$. Now, a numerical algorithm is, by definition, deterministic. It produces the same result every time we run it. How can then the obtained sequence of numbers be random? The answer is subtle: the sequence will not *be* random, but it will *look* random. Now there is a thin line between ''to look'' and ''to be'' and we ask whether ''to look'' is enough for the applications. Since the numbers we will generate ''are'' not random, but ''look'' random, they are sometimes called, for the sake of rigor, ''pseudorandom'' numbers.

Let us first explain the first historical algorithm designed to generate pseudo-random numbers. It is due to von Neumann. It takes $M = 10\,000$ and sets an initial value $m_0 \in (0, 9999)$ at will. This first choice could be deterministic (e.g., my birth year) or truly stochastic (the last four digits of the time in seconds since the computer has been up). Once the first value, the so-called seed, has been set, it is squared, m_0^2, the last two digits are discarded, and the new last four digits constitute the next value m_1. The process is repeated to obtain $m_2, m_3, \ldots$. Let us see an example taking $m_0 = 5232$ as seed.

$$\begin{aligned} m_0 = 5232 &\rightarrow m_0^2 = 27373824 \rightarrow m_1 = 3738, \\ m_1 = 3738 &\rightarrow m_1^2 = 13972644 \rightarrow m_2 = 9726, \\ m_2 = 9726 &\rightarrow m_2^2 = 94595076 \rightarrow m_3 = 5950. \end{aligned}$$

The sequence follows the algorithm

$$m_{i+1} = [m_i^2/100] \bmod 10000 \tag{A.1}$$

where $[x]$ is the integer part of x. In the previous example, this would yield the sequence of random numbers: $u_0 = 0.5232$, $u_1 = 0.3738$, $u_2 = 0.9726$, $u_3 = 0.5950$, and so on. The question is whether this sequence ''looks'' random. That is, if we give this sequence to a good statistician, will he be able to tell us that the sequence is not really random? Given a long list of numbers $(u_0, u_1, u_2, \ldots)$, what would the statistician do to determine whether it is random or not? He would apply a series of tests to check for randomness. Very sophisticated tests do exist, and it is not our intention to describe them in any detail. We just state two basic tests:

1) Moments: Check that $\langle u_i^n \rangle = (n+1)^{-1}$. In particular, $\langle u_i \rangle = 1/2$, $\langle u_i^2 \rangle = 1/3$.
2) Correlations: Check that $\langle u_i u_j \rangle = 0.25$ if $i \neq j$.

But there are many other tests that should be satisfied by true random numbers. It is not difficult to find a test that the von Neumann generator does not satisfy. For a simple reason: von Neumann generator is necessarily cyclic. That is, if there exists an integer number L such that $m_L = m_0$, then the series repeats itself $m_{i+L} = m_i$.

At most, the period L is equal to 10 000 (the maximum possible number of values for m_i), but it can be much less than that. Obviously, no truly random sequence is cyclic. Another problem is that, if the number 0 is hit at some time, $m_K = 0$, then $m_i = 0, \forall i \geq K$. It is not necessary to be a good statistician to realize that a series of zeros does not look very much random. If a modern simulation that uses millions or billions of random numbers, these problems are certain to appear sooner than later and the von Neumann algorithm, although very important conceptually, is useless from a practical point of view.

But even if von Neumann algorithm does not satisfy the minimal needs for modern simulations, it certainly sets the basic settings of what are called *congruential generators*. All we need to do is to replace M by a much larger number and devise an algorithm replacing Eq. (A.1). Before tackling this question, let us enumerate the properties we would like a good pseudorandom number generator to have.

1) *Good Statistical Properties:* This has been mentioned earlier. The generator must pass a series of statistical tests. However, we must realize that there are always some tests that the generator will fail in. The question is whether these tests are important to the particular application we are using the numbers for. For example, most generators have a period L after which they repeat themselves. This period L has to be much larger that the length of the sequence of random numbers required. A rule of thumb says that we cannot use more than $L^{1/2}$ numbers in our simulation before the good statistical properties are lost.
2) *Efficiency:* A modern simulation requires billions of numbers. Besides the generation of the random numbers, many other operations have to be made. The generation of random numbers cannot be a bottleneck for the simulation.
3) *Reproducibility:* This property might look paradoxical at first sight. How can we demand that a random number sequence be reproducible? Note, however, that we are dealing with complicated numerical algorithms that have to be fully controllable from the beginning to the end. Imagine that you rush to your thesis supervisor and tell him that your simulation has yielded a beautiful result (a critical exponent that verifies some hyperscaling theory) but that you cannot reproduce that result!! You will be in trouble. Furthermore, when we are debugging a program, we do not want to have any uncontrolled source of variability in the results of the program.

A.2 Congruential Generators

They are based on a recurrence relation of the form

$$m_{i+1} = F(m_0, m_1, \ldots, m_i) \mod M \tag{A.2}$$

and a suitable large number M. Because of the modulus[2] operation, all numbers satisfy $m_i \in [0, M-1]$. One might think that complicated nonlinear relations would

2) A modulus is also called a *congruence*, and hence the name.

be necessary to produce a "chaotic" sequence that looks random. However, it comes as a small surprise that the simplest one-step memory linear relation

$$m_{i+1} = am_i + c \quad \text{mod } M \tag{A.3}$$

is sufficient to yield random numbers with good statistical properties if the numbers a, c, and M are chosen conveniently. Notice, though, that any one-step recurrence relation of the form $m_{i+1} = F(m_i) \mod M$ will have a period L at most equal to M. Furthermore, the probability that two consecutive numbers are identical is equal to zero (otherwise the sequence repeats itself infinitely). This would not be the case if the numbers m_i were really random because then the probability that $m_{i+1} = m_i$ is equal to $1/M$. Of course, if M is large, this is a small number and it might not be important that it is instead equal to zero for our generator.

In order to stress the deterministic character of (A.3), let us write down the explicit solution of this recurrence relation:

$$m_i = \left(m_0 - \frac{c}{1-a}\right) a^i + \frac{c}{1-a} \quad \text{mod } (M). \tag{A.4}$$

The first criterion to determine whether a congruential generator defined by the set of numbers (a, c, M) has good properties is to make sure that its period L is very large. How large can it be? If $c \neq 0$, then L can be as large as M. However, if $c = 0$, we want to avoid the value $m_i = 0$, as it would lead to $m_j = 0$ if $j \geq i$. This implies that L could be at most equal to $M - 1$. In a somewhat old-fashioned terminology, a generator with $c \neq 0$ is called a "mixed" congruential generator, whereas one with $c = 0$ is called a "multiplicative" congruential generator. We only discuss the conditions under which a mixed congruential generator has the maximum period M and refer the reader to more specialized bibliography if he or she is really interested.

One can prove a theorem which says that a mixed congruential generator reaches the maximum period $L = M$ if and only if

(i) c and M are relatively prime numbers (their greatest common divisor is 1);
(ii) $a = 1 \bmod g$, for each prime divisor g of M;
(iii) $a = 1 \bmod 4$, if M is a multiple of 4.

It is very useful in a computer to use a number M of the form $M = 2^b$. This is so because computers work with binary numbers and the congruence modulus to a power of 2 is then very easy to obtain. Just think in human (base 10) terms: $12891243712340234 \mod 10^6$ is 340234, the last six digits. In the same way, for a computer it is easy to determine that $11010101111111000111 \mod 2^6 = 000111$ or, in base 10 notation, $876487 \mod 64 = 7$. In this case of $M = 2^b$, the conditions for a maximum period are simply that c must be odd and that $a = 1 \mod 4$.

Once these general properties have been established, people have searched for (in many cases using trial and error techniques) triplets of numbers (a, c, M) that yield good statistical properties. There are no theorems now, and mostly we believe what other people tell us about the quality of a generator. With this in mind, three

sets of allegedly "good" numbers are as follows:

$$\begin{array}{lll} M = 2^{35}, & a = 129, & c = 1, \\ M = 2^{32}, & a = 69069, & c = 1, \\ M = 2^{32}, & a = 1812433253, & c = 1. \end{array}$$

Although it seems that the third row of values is superior in the sense that the produced numbers have better statistical properties, the second row has been extensively used. The "deep" reason seems to be that $a = 69069$ is a number easy to remember! Most "free" random number generators (those coming built in with the machine operating system or main compilers) are congruential with $(a, c, M) = (69069, 1, 2^{32})$.

The use of $M = 2^{32}$ has another advantage because most integer arithmetic is performed with 32-bit accuracy. The maximum integer number that can be written with 32 bits is $2^{32} - 1$ and any integer multiplication whose result is larger that $2^{32} - 1$ results in an overflow. Indeed, we do not care about the overflow, since we are performing arithmetic modulus 2^{32}, and simply neglecting the overflowed bits gives the correct answer. However, compilers can get annoying if they find an overflow and it might be necessary to tell the computer not to worry about the overflow whenever it is produced (this is usually achieved with some option during compilation, e.g., `-check=nooverflow` or something similar). In this way, the modulus operation is performed automatically. The only thing to worry about is that in a 32-bit representation, a number that has the first bit, the most significant bit, equal to 1 is considered to be a negative number. In fact, integer numbers in 32-bit arithmetic run from -2^{31} to $2^{31} - 1$ while we consider them to run between 0 and $2^{32} - 1$. All we need to take into account, then, is that, if the computer tells us that the random number $u_i = m_i/M$ is negative, indeed it should be $(m_i + M)/M = 1 + u_i$.

With all this in mind, let us give a simple computer program for the function `ran_u(m)` that generates pseudorandom numbers uniformly distributed in the $(0, 1)$ interval.

```
function ran_u()
integer, save :: m=1234567
parameter (rm=2.0**(-32),ia=1812433253,ic=1)
m=m*ia+ic
ran_u=rm*m
if (ran_u < 0.0) ran_u=1.0+ran_u
end function ran_u
```

It uses the particular value `m=1234567` as the seed. If we want a different sequence of numbers, we can change the value for the seed. For this program to work, we have to be sure that the integer arithmetic is performed with 32-bit precision and that the program does not detect overflows. We can avoid these conditions if we use double precision arithmetic, such as

```
double precision function ran_u()
double precision ia,ic,ma,rm
double precision, save :: m=1234567.0d0
parameter (ma=2.0d0**32,rm=2.0d0**(-32))
parameter (ia=1812433253.0d0, ic=1.0d0)

m=mod(m*ia+ic,ma)
ran_u=rm*m,

end function ran_u
```

but this is usually slower that the previous implementation. Do not forget to define in this case `ran_u` as double precision in the calling program.

A.3 A Theorem by Marsaglia

We have already mentioned that numerical deterministic algorithms will not produce true random numbers and that there will always be some test that our generator will not pass. Here comes an interesting test that a simple congruential random generator does not pass. Let us take the recurrence relation (A.3) with $M = 2^8$, $a = 25$, $c = 1$. We generate random numbers $(u_0, u_1, u_2, \ldots)$ and organize them in pairs (u_i, u_{i+1}). Each pair is then plotted in a two-dimensional plane. The result is shown in Figure A.1(a). If the numbers u_i were truly independent of each other, the points would look uniformly distributed in the plane, while it is clear that there is an underlying pattern. The first reaction is that we should blame our choice of the numbers (a, c, M), which was not good enough. Indeed, if we repeat the plot using the numbers $(69069, 1, 2^{32})$, the result looks much better, see the right panel of Figure A.1. Is there an *a priori* way of choosing the triplet (a, c, M) such that these correlations between consecutive numbers do not exist?

A surprising and negative answer was given by Marsaglia in a very interesting article [63] with the suggestive title *Random Numbers Fall Mainly in the Planes*. In short, Marsaglia has shown that *all* congruential generators will have subtle correlations. These correlations show up when we organize the sequence of random numbers $(u_0, u_1, u_2, \ldots)$ in groups of d to generate points in a d-dimensional space $z_1 = (u_0, u_1, \ldots, u_{d-1})$, $z_2 = (u_d, u_{d+1}, \ldots, u_{2d-1})$, and so on. The theorem proves that there exists a dimension d for which all points $z_1, z_2, \ldots$ fall on a hyperplane of dimension $d - 1$. This is rather annoying and shows that congruential generators fail to pass a particular test for independence. The accepted compromise is to use a generator that has a large dimension d for the Marsaglia planes. Only then can we accept that the presence of these correlations between the numbers will not hopefully be of any significance for our calculation.

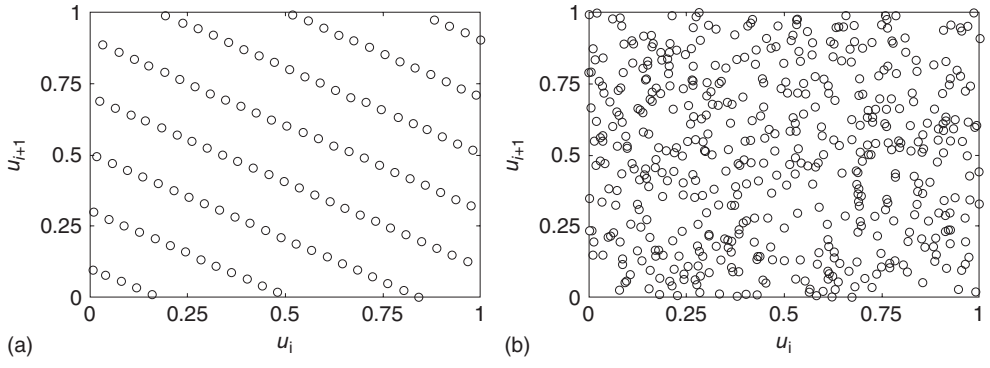


Figure A.1 Plots of pairs of random numbers (u_i, u_{i+1}) obtained by using the congruential algorithm (A.3) with $(a, c, M) = (25, 1, 2^8)$ (a) and $(a, c, M) = (69069, 1, 2^{32})$ (b). Note the clear presence of the Marsaglia planes in this $d = 2$ plot in panel (a).

A.4 Feedback Shift Register Generators

"Feedback Shift Register" (FSR) random number generators use similar ideas to the congruential generators but organize the resulting numbers in a different way. The idea is to use a congruential generator modulus 2 (so it only produces 0 and 1) but to increase the memory of the recurrence relation. If we denote by z_i the different bits, the recurrence relation is

$$z_i = c_1 z_{i-1} + c_2 z_{i-2} + \cdots + c_p z_{i-p} \mod 2 \tag{A.5}$$

where $c_k = 0, 1$ for $k = 1, \ldots, p$, is a given set of binary constants and we need to set the initial values $(z_0, z_1, \ldots, z_{p-1})$. The actual random numbers are obtained first by constructing b-bit integers by joining the bits $m_0 = z_0 z_1 \cdots z_{b-1}$, $m_1 = z_b z_{b+1} \cdots z_{2b-1}$, and so on. The real numbers are, as before, $u_i = m_i / 2^b$.

It is clear from the recurrence relation that the numbers z_i necessarily repeat after a maximum period of $L = 2^p$ and hence the maximum available number of distinct random numbers is 2^{p-b}. How can we obtain a maximum period? A theorem tells us that an almost maximum period of $2^p - 1$ is obtained in the recurrence relation (A.5) if and only if the polynomial

$$f(z) = 1 + c_1 z + c_2 z^2 + \cdots + c_p z^p \tag{A.6}$$

cannot be factorized as $f(z) = f_1(z) f_2(z)$ (all operations are modulus 2). Technically, $f(z)$ is said to be primitive in GF(2), the Gaulois field. It turns out that good statistical properties can be obtained by using the simplest primitive polynomials, those of the form

$$f(z) = 1 + z^q + z^p \tag{A.7}$$

with suitable values for p and q. The recurrence relation becomes

$$z_i = z_{i-p} + z_{i-q} \mod (2). \tag{A.8}$$

If, instead of binary arithmetic, we use binary logical operations, this relation is equivalent to

$$z_i = z_{i-p} \otimes z_{i-q} \tag{A.9}$$

where $\otimes$ is the *exclusive or* logical operation (let us recall its table: $0 \otimes 0 = 0, 0 \otimes 1 = 1, 0 \otimes 1 = 1, 1 \otimes 1 = 0$). The exclusive or operation is included in many compilers and it has the advantage that it is really fast.

A possible way to implement the algorithm is to work directly at the level of the m_i integers. First, set p initial values $m_0, m_1, \ldots, m_{p-1}$, where m_i are integers with the desired accuracy (for instance, $b = 31$ bits, so we avoid the presence of negative numbers). This is implemented using another random generator. Then use the recurrence relation $m_i = m_{i-p} \otimes m_{i-q}$, which is understood by the compiler as the operation $\otimes$ performed at the level of each bit of the integer numbers. Finally, set $u_i = m_i/2^b$.

Once the maximum period has been ensured, which pairs (p, q) will give good statistical properties? A first suggested choice was $p = 250, q = 103$. The resulting generator is called R250. Its period is $2^{250} \approx 1.8 \times 10^{75}$, which is sufficiently large for any application. It was very popular for a time until it was shown in [64] that some spurious correlations showed up when computing the equilibrium properties of the Ising model in regular lattices, probably due to the existence of Marsaglia planes of not large enough dimension. Other values that have been suggested as producing good statistical properties are $p = 1279$, $q = 418$.

A.5
RCARRY and Lagged Fibonacci Generators

The family of general lagged Fibonacci generators uses the relation

$$m_i = (m_{i-p} \odot m_{i-q}) \mod M \tag{A.10}$$

where $\odot$ denotes a sum, a subtraction, a multiplication, or an exclusive or (performed bit by bit, in this case we recover the FSR generators). When using a sum or a subtraction, there is a mixing of bits and, allegedly, the statistical properties are better.

James [65], based on the ideas of Marsaglia *et al.* [66], proposed the so-called RCARRY generator. It starts from a lagged Fibonacci sequence with a subtraction but adding an additional subtraction if m_i is a negative number. The algorithm is

$$m_i = m_{i-r} - m_{i-s} - c_i \mod (M)$$

where $r > s$ and the remainder is $c_i = 1$ if $m_i \leq 0$ (before the modulus operation) and $c_i = 0$ otherwise. The subtraction mixes the different bits of the integer number m_i, and the use of the remainder c_i is meant to destroy most of the correlations in the sequence of random numbers. For the initialization of the algorithm, one needs to give a sequence of r integer number m_i, $i = 1, \ldots, r$. A convenient choice is $M = 2^{24}$, $r = 24, s = 10$. The period of the generator is 48 times less than the

number of different states that can be represented using 24 numbers with 24 bits, or $(2^{24})^{24} \approx 10^{173}$.

A.6 Final Advice

The reader might be optimistic by nature and conclude that there are some good random numbers that are better than others. Or he might be pessimistic and conclude that all random number generators are bad, and some of them are worse than others. There are standard batteries of statistical tests to check the suitability of random number generators such as the one developed at NIST [67]. The truth is that a pseudorandom number generator that uses a deterministic algorithm will never pass all the tests that check the goodness of the generator. The most important point, in our opinion, is not to rely blindly on the random number generator provided by ''we do not know who'' and use only the generators considered to be ''good'' in the literature. But be aware that even those good generators might have problems. If a weird result is found in your simulations, it might be worthwhile to check that it does not have its origin in the random number generator, for instance, by repeating the simulation with a different generator.

Exercises

1) Program von Neumann's algorithm. Generate 1000 numbers using this algorithm and compute the average values $\langle x \rangle$, $\langle x^2 \rangle$, and the correlation $\langle x_i x_{i+1} \rangle$. Do results depend on the seed m_0?
2) Use the random number provided by your favorite compiler or library and check whether the period is larger than $2^{32} \approx 4.3 \times 10^9$. Generate 10^6 numbers using this algorithm and compute the average values $\langle x \rangle$, $\langle x^2 \rangle$, and the correlation $\langle x_i x_{i+1} \rangle$. If the generator requires a seed, do the results depend on the seed m_0?
3) Repeat the previous problem using l'Ecuyer's algorithm RANECU, which uses $a = 40692$, $m = 2147483399$, $c = 0$ (purely multiplicative).
4) Repeat the same with algorithm R250.
5) Compute numerically the dimension of Marsaglia's planes for R250 and the congruential algorithm with $M = 2^{32}$, $a = 69069$, $c = 1$.
6) A famous problem states that the probability that the second degree equation $ax^2 + bx + c = 0$ has real roots is $\frac{5}{36} + \frac{\log(2)}{6} \approx 0.254$ if a, b, c have been chosen randomly and uniformly in the interval $(0, 1)$. Use your favorite random number generator to check this result. Do not forget to include the errors of the estimator of this probability.

Appendix B
Generation of *n*-Dimensional Correlated Gaussian Variables

We have already explained in Section 3.5 a method valid for the generation of a set of random variables $x = (x_1, x_2, \dots, x_n)$ with a jointly Gaussian distribution given by (1.80) with mean values $\mu = (\mu_1, \dots, \mu_n)$ and correlation matrix $C_{ij} = \langle(\hat{\mathbf{x}}_i - \mu_i)(\hat{\mathbf{x}}_j - \mu_j)\rangle$ with $C = A^{-1}$. An equivalent way of looking at this problem is via the diagonalization of the quadratic form in the exponential of (1.80). This means the use of the matrix relation $A = \Phi^{\intercal} D \Phi$, with $\Phi^{\intercal}$ being the transpose matrix of Φ, the matrix of change of variables, and D a diagonal matrix with diagonal elements (the eigenvalues) $(\lambda_1, \lambda_2, \dots, \lambda_n)$. As A is a symmetric matrix, this diagonalization is always possible and, as the quadratic form is supposed to be positive definite, the eigenvalues are strictly positive, $\lambda_i > 0, \forall i$. Furthermore, the determinant of the matrix of the change of variables is $|\Phi| = 1$ and $|A| = \prod_{i=1}^{n} \lambda_i$. The change of variables $x = \mu + \Phi z$ or $x_i = \mu_i + \sum_{j=1}^{n} \Phi_{ij} z_j$, in coordinates, changes the quadratic form to $\exp\left[-\frac{1}{2} z^{\intercal} D z\right] = \exp\left[-\frac{1}{2} \sum_{i=1}^{n} \lambda_i z_i^2\right]$. As the Jacobian of this change is $|\Phi| = 1$, it leads to a pdf for the z variables

$$f_{\hat{\mathbf{z}}_1, \dots, \hat{\mathbf{z}}_n}(z_1, z_2, \dots, z_n) = \sqrt{\frac{\prod_{i=1}^{n} \lambda_i}{(2\pi)^n}} \exp\left[-\frac{1}{2} \sum_{i=1}^{n} \lambda_i z_i^2\right] \tag{B.1}$$

$$= \prod_{i=1}^{n} \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left[-\frac{z_i^2}{2\sigma_i^2}\right] \tag{B.2}$$

indicating that the z_i's are independent Gaussian variables of zero mean and variance $\sigma_i^2 \equiv 1/\lambda_i$. Once the set of z_i's has been generated by our favorite Gaussian number generator, we just need to change back the variables to the x_i's. In general, this algorithm is slow (it requires the solution of a full matrix diagonalization problem, finding the eigenvalues and the eigenvectors) but it can be an alternative to the one explained in Section 3.5. It turns out that there are some cases of interest where the change of variables adopts a particularly simple form and the whole algorithm can be speeded up with the help of the fast Fourier transform. We now have a look at two of these cases.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

B.1 The Gaussian Free Model

Let us here consider the so-called 1-D free model, in which the Gaussian joint pdf has the particular expression

$$f_{\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_N}(x_1,\ldots,x_n) = C\exp[-\mathcal{L}_0(x_1,\ldots,x_n)] \tag{B.3}$$

where the quadratic form given by the function $\mathcal{L}_0$ is[1]:

$$\mathcal{L}_0(x_1,\ldots,x_n) = \frac{1}{2}\sum_{i=1}^{n}\left[(x_{i+1}-x_i)^2 + ax_i^2\right], \quad a>0 \tag{B.4}$$

and C is the normalization constant. Here we consider what are called ''periodic boundary conditions,'' which have been discussed previously: that is, whenever x_{n+1} appears in any formula, it should be replaced by x_1. If we write the quadratic form as $\mathcal{L}_0 = \frac{1}{2}x^{\intercal}Ax$, matrix A is

$$A = \begin{pmatrix} 2+a & -1 & 0 & 0 & \cdots & 0 & -1 \\ -1 & 2+a & -1 & 0 & \cdots & 0 & 0 \\ 0 & -1 & 2+a & -1 & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & 0 & 0 & \cdots & -1 & 0 \\ 0 & 0 & 0 & 0 & \cdots & a+2 & -1 \\ -1 & 0 & 0 & 0 & \cdots & -1 & a+2 \end{pmatrix} \tag{B.5}$$

with diagonal elements $A_{ii} = a+2$, off-diagonal $A_{i-1,i} = A_{i,i-1} = -1$, and the corner elements $A_{1,n} = A_{n,1} = -1$.

We make now a change of variables based on the discrete Fourier transform of the variables[2]. Let us define, then, a set of complex variables $\hat{x}_0,\ldots,\hat{x}_{n-1}$ as

$$\hat{x}_k = \sum_{j=0}^{n-1} e^{\frac{2\pi \mathbf{i}}{n}jk}x_{j+1}, \qquad k=0,1,\ldots,n-1. \tag{B.6}$$

Here, $\mathbf{i}$ stands for the imaginary unit $\mathbf{i} = \sqrt{-1}$. The inverse transformation is

$$x_{j+1} = \frac{1}{n}\sum_{k=0}^{n-1} e^{-\frac{2\pi \mathbf{i}}{n}jk}\hat{x}_k, \qquad j=0,\ldots,n-1. \tag{B.7}$$

Note that the definition (B.6) can be used for whatever value of $k\in\mathbb{Z}$, but as $\hat{x}_{n+k} = \hat{x}_k$, only values of $\hat{x}_k$ between $k=0$ and $k=n-1$ are, in general, independent of each other. In the particular case where the x_j's are real variables, it follows the additional condition $\hat{x}_{-k} = \hat{x}_{n-k} = \hat{x}_k^*$. This reduces the number of independent values of $\hat{x}_k$ even further. The analysis of the independent set of $\hat{x}_k$ is slightly

1) In some contexts, this function is called a ''free Lagrangian.''
2) The reader not familiar with the Fourier transform might have a look at Appendix G. For convenience, the definitions used in this section are slightly different from those of the Appendix, where the numbers x_i run from $i=0$ to $i=N-1$, whereas in the notation used here they run from $i=1$ to $i=n$.

different for n even or n odd. In what follows, we consider only the case where n is an even number and leave the reader to redo the details for the case of n odd. For instance, if $n = 8$, the following relations apply: $\hat{x}_7 = \hat{x}_1^*$, $\hat{x}_6 = \hat{x}_2^*$, $\hat{x}_5 = \hat{x}_3^*$, as well as $\hat{x}_0 = \hat{x}_0^*$, $\hat{x}_4 = \hat{x}_4^*$. These relations imply that, if we write $\hat{x}_k = r_k + is_k$, then $s_0 = s_4 = 0$ and the only independent variables are $r_0, r_1, s_1, r_2, s_2, r_3, s_3, r_4$. For compactness, we will use the notation $(z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8) \equiv (r_0, r_4, r_1, s_1, r_2, s_2, r_3, s_3)$ and a similar generalization for arbitrary n even.[3)]

The nice feature of the change of variables (B.6) and (B.7) is that it diagonalizes the quadratic form

$$\sum_{i=1}^{n} \left[(x_{i+1} - x_i)^2 + a x_i^2\right] = \frac{1}{n} \sum_{k=0}^{n-1} \omega_k^2 |\hat{x}_k|^2, \qquad \omega_k^2 = a + 4 \sin^2\left(\frac{\pi k}{n}\right). \tag{B.8}$$

To prove this identity, it is useful to have the relation

$$\frac{1}{n} \sum_{i=1}^{n} \mathrm{e}^{\frac{2\pi \mathrm{i}}{n} ik} = \begin{cases} 1, & k = 0, \\ 0, & k \neq 0. \end{cases} \tag{B.9}$$

However, we must not look at $\mathcal{L}_0$ written in terms of $\hat{x}_k$ and jump to the conclusion that the full set of the $\hat{x}_k$'s are independent Gaussian variables, because we just proved that there are relations between them and some of these variables depend on the others. We need to rewrite the quadratic form $\mathcal{L}_0$ using only the set of independent variables $z_k, k = 1, \ldots, n$. Including only these variables and using $\omega_{n-k} = \omega_k$, the quadratic form can be written as[4)]

$$\begin{aligned} \frac{1}{n} \sum_{k=0}^{n-1} \omega_k^2 |\hat{x}_k|^2 &= \frac{1}{n} \left(\omega_0^2 |\hat{x}_0|^2 + \omega_{\frac{n}{2}}^2 r_{\frac{n}{2}}^2 + 2 \sum_{k=1}^{\frac{n}{2}-1} \omega_k^2 |\hat{x}_k|^2 \right) \\ &= \sum_{k=1}^{n} \Lambda_k z_k^2 \end{aligned} \tag{B.10}$$

with $\Lambda_1 = \omega_0^2/n, \Lambda_2 = \omega_{\frac{n}{2}}^2/n, \Lambda_3 = \Lambda_4 = 2\omega_1^2/n, \Lambda_5 = \Lambda_6 = 2\omega_2^2/n$, and so on. As $f_{\hat{\mathbf{z}}_1, \ldots, \hat{\mathbf{z}}_n}(z_1, \ldots, z_n) \propto e^{-\frac{1}{2} \sum \Lambda_k z_k^2}$, this shows that the z_k's are independent Gaussian variables of zero mean and variance $\sigma_k^2 = 1/\Lambda_k$. Once these variables have been generated, we use the inverse discrete Fourier transform (B.7) to obtain the original variables $(x_1, \ldots, x_n)$.

To sum up, to generate n Gaussian variables $(x_1, \ldots, x_n)$ whose pdf is (B.3), we generate a set of independent Gaussian variables $(z_1, \ldots, z_n)$ of zero mean and variance $\sigma_k^2 = 1/\Lambda_k$. From these variables, we construct the Fourier variables $(\hat{x}_1, \ldots, \hat{x}_n)$ fulfilling the symmetry relations $\hat{x}_{-k} = \hat{x}_{n-k} = \hat{x}_k^*$. Finally, using an inverse discrete Fourier transform, we obtain the desired set $(x_1, \ldots, x_n)$.

In practice, most fast Fourier transform routines organize their variables such that, for the discrete Fourier transform of a set of real variables x_i, only the

3) For $n = 9$, an odd number, the relations between the Fourier variables is $\hat{x}_8 = \hat{x}_1^*$, $\hat{x}_7 = \hat{x}_2^*$, $\hat{x}_6 = \hat{x}_3^*$, $\hat{x}_5 = \hat{x}_4^*$, and $\hat{x}_0 = \hat{x}_0^*$. The set of independent Fourier variables is $(z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9) = (r_0, r_1, s_1, r_2, s_2, r_3, s_3, r_4, s_4)$.

4) The reader can check this formula using $n = 8$, for example.

independent set of values z_k are kept in memory, precisely in the same order that have been defined here. For example, if $(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8)$ is the original vector of $n = 8$ real variables, the Fourier coefficients $\hat{x}_k = r_k + is_k$, are kept in a vector organized precisely as $(r_0, r_4, r_1, s_1, r_2, s_2, r_3, s_3)$.[5] The next program listing implements this algorithm.

```
program freeGaussian
implicit double precision (a-h,o-z)
parameter (n=256)
dimension x(n)
open(66,file='freeGaussian.dat',status='unknown')
a=1.0d0
pin=3.14159265358979d0/n

   x(1)=ran_g()*dsqrt(n/a)
   x(2)=ran_g()*dsqrt(n/(a+4.0d0))
   do i=2,n/2
     sigma=dsqrt(n/(2.0d0*(a+4.0d0*(dsin(pin*(i-1)))**2)))
     x(2*i-1)=sigma*ran_g()
     x(2*i)=  sigma*ran_g()
   enddo
   call realfft1d(x,n,-1)

do i=1,n
  write(66,*) x(i)
enddo
end program freeGaussian
```

Note that, in our notation, `x(2)` contains really $r_{n/2}$ or z_2. We have used the values $\Lambda_1 = \omega_0^2/n = a/n$, $\Lambda_2 = \omega_{\frac{n}{2}}^2/n = (a+4)/n$. We include the call to `realfft1d(x,n,-1)`, a generic name for a routine that provides the inverse discrete Fourier transform in the case where the variables are real numbers.[6]

B.2 Translational Invariance

The second case[7] in which the use of discrete Fourier transform can be useful is when the correlation function of the random variables $(x_1, \ldots, x_n)$ depends only on the absolute value of the difference between the indexes, that is, when $\langle (x_i - \mu_i)(x_j - \mu_j) \rangle = C_{ij} = C_{|i-j|}$, $i, j = 1, n$. As we will see, the method we are going to explain yields a correlation function that satisfies, beyond the general condition $C_\ell = C_{-\ell}$, the so-called periodic boundary conditions, namely $C_\ell = C_{n+\ell} = C_{n-\ell}$. These boundary conditions appear naturally if we consider that the $(x_1, \ldots, x_n)$

5) Or, in general, organized as $(r_0, r_{n/2}, r_1, s_1, r_2, s_2, \ldots, r_{n/2-1}, s_{n/2-1})$.
6) Remember to read the exact definition of the discrete Fourier transform used by the numerical fast Fourier transform package in order to correct by the right factors of n if needed, see Appendix G.
7) Actually, this second case includes the first one as a particular example.

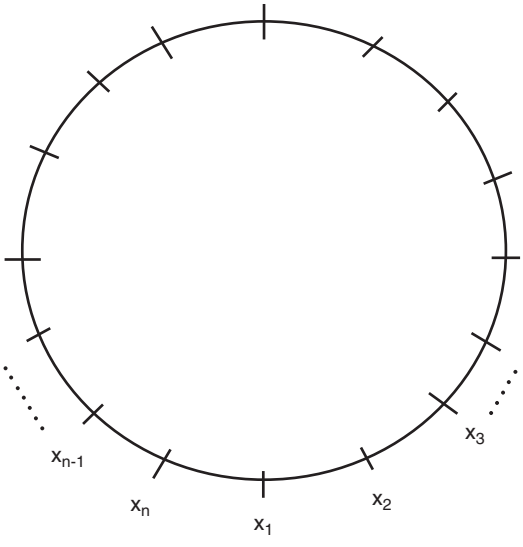


Figure B.1 Graphical representation of the periodic boundary conditions for the set of variables $(x_1, \dots, x_{2n})$.

variables are placed in a ring such that x_1 and x_n are neighbors of each other, see Figure B.1. In this setup, and considering $n = 8$ for the sake of clarity, it is clear that the correlation between the x_1 and x_3 variables is the same whether we consider that the distance separating them is $\ell = 3 - 1 = 2$ if computed counterclockwise from x_1 to x_3, or $\ell = 8 + 1 - 3 = 6$, computed clockwise, hence it is natural to assume that $C_2 = C_{8-2} = C_6$. If it is not reasonable to assume that our set of random variables $(x_1, \dots, x_n)$ fulfill the periodic boundary conditions, there is a simple trick we can use. We first double the number of variables and consider $(x_1, \dots, x_{2n})$, which we then generate assuming periodic boundary conditions. Finally, simply discard all variables $(x_{n+1}, \dots, x_{2n})$, leaving us with the required set $(x_1, \dots, x_n)$.[8)]

Without loss of generality, we assume $\mu_i = 0$ and hence $C_{|i-j|} = \langle x_i x_j \rangle$. When $\mu_i \neq 0$, all we need to do is to add μ_i to the generated value of the random variable x_i. Let $\hat{x}_k$ be the discrete Fourier transform of x_i as defined in (B.6). As x_j are real numbers, it is $\hat{x}_k^* = \hat{x}_{-k}$, from which

$$\begin{aligned} \langle |\hat{x}_k|^2 \rangle = \langle \hat{x}_k \hat{x}_{-k} \rangle &= \sum_{i,j=0}^{n-1} e^{\frac{2\pi i}{n}(i-j)k} \langle x_{i+1} x_{j+1} \rangle \\ &= \sum_{i,j=0}^{n-1} e^{\frac{2\pi i}{n}(i-j)k} C_{|i-j|}. \end{aligned} \tag{B.11}$$

If we now make the change of variables $\ell = i - j$ and use the periodic boundary conditions $C_\ell = C_{n+\ell}$, we arrive at

8) The variables $(x_{n+1}, \dots, x_{2n})$ also satisfy the required correlations, but they are not independent of the set $(x_1, \dots, x_n)$.

$$\frac{1}{n}\langle|\hat{x}_k|^2\rangle = \sum_{\ell=0}^{n-1} e^{\frac{2\pi i}{n}\ell k} C_\ell \equiv S_k. \tag{B.12}$$

We have defined S_k, the discrete Fourier transform of the correlation function C_ℓ. This function is called, in some contexts, the "structure factor," or the "power spectrum" in others. It is obvious from its definition that it is a real, positive function. Moreover, it satisfies $S_{-k} = S_k$and $S_{k+n} = S_k$. It is possible to invert the discrete Fourier transform to find C_ℓ from S_k:

$$C_\ell = \frac{1}{n}\sum_{k=0}^{n-1} e^{-\frac{2\pi i}{n}\ell k} S_k, \qquad \ell = 0, \dots, n-1. \tag{B.13}$$

We now introduce a set of independent Gaussian variables $(z_1, \dots, z_n)$ of zero mean and variance 1, $\langle z_i z_j\rangle = \delta_{ij}$. It is possible to show that, if $\hat{z}_k$ is the discrete Fourier transform, defined as in (B.6), the corresponding structure factor turns out to be $\langle|\hat{z}_k|^2\rangle = n$. Therefore, if we define $\hat{x}_k = S_k^{1/2}\hat{z}_k$, it is clear that $\frac{1}{n}\langle|\hat{x}_k|^2\rangle = S(k)$, the desired relation (B.12) in Fourier space. All that remains now is to use the inverse discrete Fourier transform (B.7) to obtain $(x_1, \dots, x_n)$.

In summary, to generate n values $(x_1, \dots, x_n)$ of Gaussian random variables with a given correlation function C_ℓ, follow the next steps:

(0) Compute the discrete Fourier transform S_k of the correlation function C_ℓ (implemented using periodic boundary conditions).
(1) Generate a set of independent Gaussian variables $(z_1, \dots, z_n)$ of zero mean and variance 1.
(2) Compute its discrete Fourier transform $\hat{z}_k = \mathcal{F}_D(z)$ and obtain $\hat{x}_k = S_k^{1/2}\hat{z}_k$.
(3) Compute the inverse discrete Fourier transform to obtain $x = \mathcal{F}_D^{-1}(\hat{x})$.
(4) Add the average value μ_i to x_i, if needed.

This method is known in the literature as the *Fourier filtering method* [68]. A final word of warning is necessary here. Many times, one introduces an *ad hoc* correlation function C_ℓ, for instance, a power-law type, $C_\ell \sim |\ell|^{-\gamma}$. To use the previous algorithm, we need precise values for C_ℓ for $\ell = 0, \dots, n-1$ and the fulfillment of the periodic boundary conditions $C_\ell = C_{n-\ell}$. Therefore, it is important to be more precise about the exact meaning of "$C_\ell \sim |\ell|^{-\gamma}$." For example, we could define (we assume n even)

$$C_\ell = \begin{cases} C_0, & \ell = 0, \\ |\ell|^{-\gamma}, & -n/2 \le \ell \le n/2, \ell \neq 0 \end{cases} \tag{B.14}$$

with a given value for C_0, supplemented with the periodic boundary conditions $C_{n-\ell} = C_{-\ell} = C_\ell$, whenever needed. One has to be careful, though, with the value for C_0. If one computes, in general numerically using (B.12), the structure factor S_k corresponding to this definition, one will notice that it can become negative for some ranges of values of k depending on the value of C_0. This means that (B.14) is such that the quadratic form $\sum_{i,j} x_i A_{i,j} x_j$ with $(A^{-1})_{i,j} = C_{|i-j|}$ is not positive definite and, therefore, *there does not exist a set of Gaussian random variables whose correlation function is* (B.14) *with this particular choice for* C_0.

This example shows that the first thing we have to do when confronted with a "reasonable" correlation function C_ℓ is to check whether the associated quadratic form $\sum_{i,j} x_i A_{i,j} x_j$ is positive definite. This is done by checking that all elements of the discrete Fourier transform S_k are positive. If that is not the case, one can either discard the given C_ℓ or "fix it." The simplest way of fixing it is by finding the minimum (negative) value $S_{\min} = \min_k S_k$, and then replacing $S_k \to S_k - S_{\min}$. This, of course, ensures that $S_k \geq 0, \forall k$ and, at the level of the correlation function, all this procedure does is to replace C_0 by a new value while keeping unchanged all other values of $C_\ell, \ell \neq 0$. We call this the "minimal subtraction procedure."

One could use, instead, the following expression for the correlation function:

$$C_\ell = (1 + \ell^2)^{-\gamma/2}, \qquad -n/2 \leq \ell \leq n/2 \tag{B.15}$$

supplemented, again, with the periodic boundary conditions $C_{n-\ell} = C_{-\ell} = C_\ell$, whenever needed. In this case, the resulting S_k is always positive and we do not need to worry about any modifications.[9] We now present a program listing that implements this algorithm for the generation of Gaussian random numbers with a power-law correlation function of the form (B.15).

```
program C1d
implicit double precision (a-h,o-z)
parameter(N=64)
double complex s(0:N-1), x(0:N-1)
cc(i)=1.0d0/(1.0d0+i**2)**3.0d0 gamma/2
s(0)=dcmplx(cc(0),0.0d0)
do i=1,N/2
  s(i)=dcmplx(cc(i),0.0d0)
  s(N-i)=s(i)
enddo
call fft1d(s,N,1)
ss=0.0d0
do i=0,N-1
  ss=min(ss,real(s(i)))
enddo
if (ss < 0.0d0) stop 'Non-positive quadratic form'
s=dsqrt(real(s))
!Actual generation begins here
```

```
do i=0,N-1
  x(i)=ran_g()
enddo
call fft1d(x,N,1)
x=x*s
call fft1d(x,N,-1)
```

```
end program C1d
```

Note that the first lines up to the boxed lines are needed (i) to check that the correlation value is acceptable as it yields a positive definite quadratic form, and

9) This is true if $\gamma > 0$. Negative values of γ and small values of n could again yield negative S_k and require again the use of the minimal subtraction procedure.

(ii) to generate the discrete Fourier transform S_k and its square root $S_k^{1/2}$ from the given correlation function C_i defined as `cc(i)`. These lines need to be run only once. The actual generation of the x_i's starts by the assignation of Gaussian values to the components `z(i)`. There are some symmetries that can be used to simplify this program using the fact that S_k is actually a real number. Note that the desired random numbers $x_1, \ldots, x_n$ are stored in a complex vector `x(0:N-1)` but, in fact, the imaginary part of any component `x(i)` is equal to zero.[10)]

There are other ways in which a power-law-type $C_\ell \sim |\ell|^{-\gamma}$ can be accomplished. A possibility is to take as a starting point directly S_k instead of C_ℓ using some (reasonable) criterion. For instance, take (B.15) not as the real correlation function but as a starting point and define the structure factor suggested by the manipulation

$$S_k = \sum_{\ell=-\frac{n}{2}+1}^{\frac{n}{2}} e^{\frac{2\pi i}{n}\ell k} C_\ell \approx \int_{-\infty}^{\infty} d\ell\, e^{iq\ell} C_\ell = \int_{-\infty}^{\infty} d\ell\, \frac{e^{iq\ell}}{(1+\ell^2)^{\gamma/2}} \tag{B.16}$$

$$= \frac{2\pi^{1/2}\left(\frac{|q|}{2}\right)^{\frac{\gamma-1}{2}} K_{\frac{\gamma-1}{2}}(|q|)}{\Gamma\left(\frac{\gamma}{2}\right)} \tag{B.17}$$

with $q \equiv \frac{2\pi}{n}k$ and $K_\nu(z)$ is the modified Bessel function of the second kind. We then **define**

$$S_k = \frac{2\pi^{1/2}\left(\frac{\pi|k|}{n}\right)^{\frac{\gamma-1}{2}} K_{\frac{\gamma-1}{2}}\left(\frac{2\pi|k|}{n}\right)}{\Gamma\left(\frac{\gamma}{2}\right)} \tag{B.18}$$

which satisfies $S_k \geq 0, \forall k$. Now, we can skip step (0) and take this expression for S_k as the starting point for the algorithm. The resulting correlation function will not be given *exactly* by (B.15) but will still have the same asymptotic behavior $C_\ell \sim \ell^{-\gamma}$. This was essentially the procedure used in [69].

Exercises

1) Prove (B.8).
2) Prove (B.12).
3) Run the program to generate n Gaussian variables distributed according to the $1-D$ free Lagrangian, compute the correlation function of the resulting numbers, and compare with the exact result

$$C_\ell = \frac{\sinh((n-\ell)x) + \sinh(\ell x)}{4\sinh(x)\sinh^2\left(\frac{nx}{2}\right)}$$

with $x = \operatorname{argcosh}(1 + a/2)$.

10) Typically, the imaginary part of `x(i)` turns out to be a very small number, of the order of 10^{-12} or less, due to rounding-off errors of the fast Fourier transform routines.

4) Take the correlation function C_ℓ of the previous exercise as the starting point and use the method based on the translational invariance property to generate values of Gaussian random variables distributed according to the 1-D free Lagrangian.
5) Compare the efficiency of the general method explained in Section 3.5 and the one based on Fourier transforms to generate n Gaussian variables distributed according to the 1-D free Lagrangian.
6) Generalization to d dimensions: Consider a set of $n = L^d$ variables labeled as $x_{i_1,i_2,\ldots,i_d}$ with $i_k = 1, \ldots, L$, such that the exponent of the quadratic form is

$$\mathcal{L}_0(x_1, \ldots, x_n) = \frac{1}{2} \sum_{i_1=1}^{L} \cdots \sum_{i_d=1}^{L} \left[\sum_{\mu=1}^{d} (x_{\vec{i}_\mu} - x_{\vec{i}})^2 + a x_{\vec{i}}^2 \right], \quad a > 0$$

where we have introduced the notation $\vec{i} = (i_1, \ldots, i_d)$ and $\vec{i}_\mu = (i_1, \ldots, i_\mu + 1, \ldots, i_d)$. Write down a program to generate Gaussian variables distributed according to the free Lagrangian in d dimensions.
7) Prove that, if $(x_1, \ldots, x_n)$ is a set of independent Gaussian variables of mean zero and variance 1, then its discrete Fourier transform $\hat{x}_k$ satisfies $\langle |\hat{x}_k|^2 \rangle = n$ for $k = 0, \ldots, n-1$.
8) Show that for the correlation function (B.14), the minimum value of S_k occurs at $k = n/2$, and that, in order to keep the quadratic form $\sum_{i,j} x_i A_{i,j} x_j$ positive definite, C_0 must satisfy

$$C_0 \geq (-1)^{n/2} \left(\frac{2}{n}\right)^\gamma - 2 \sum_{\ell=1}^{n/2} \frac{(-1)^\ell}{\ell^\gamma}.$$

For large n, this tends to $2(1 - 2^{1-\gamma})\zeta(\gamma)$, with $\zeta(\gamma)$ being the Riemann zeta function.
9) Use (B.18) to generate Gaussian random numbers as explained in the text, and check that the resulting correlation function tends asymptotically to a power law with exponent $-\gamma$.

Appendix C
Calculation of the Correlation Function of a Series

Let us consider a series of (real) numbers G_i, $i = 1, \ldots, M$. They could be numbers coming out from measurements in a Monte Carlo simulation or any other source. We want to determine its normalized correlation function

$$\rho_G(j) = \frac{\langle G_i G_{i+j}\rangle - \langle G\rangle^2}{\langle G^2\rangle - \langle G\rangle^2} \tag{C.1}$$

with $\langle G\rangle = \frac{1}{M}\sum_{i=1}^{M} G_i$, and $\langle G^2\rangle = \frac{1}{M}\sum_{i=1}^{M} G_i^2$. Note that, by definition, $\rho_G(0) = 1$. We assume that the series is stationary; this means that the correlation function $\rho_G(j)$ defined above does not depend on i. We first make a straightforward simplification. If we define

$$z_i = \frac{G_i - \langle G\rangle}{\sqrt{\langle G^2\rangle - \langle G\rangle^2}} \tag{C.2}$$

it can be easily proved that the correlation function $\rho_G(j)$ of the series G_i is equal to that of the series z_i:

$$\rho_z(j) = \langle z_i z_{i+j}\rangle. \tag{C.3}$$

As we have assumed that the series is stationary, we can compute this average directly including all possible values of i. So, to compute $\rho_z(1)$ we would include the $M-1$ contributions $z_1z_2 + z_2z_3 + \cdots + z_{M-1}z_M$, to compute $\rho_z(2)$ we would include the $M-2$ contributions $z_1z_3 + z_2z_4 + \cdots + z_{M-2}z_M$, and so on. In general, we have

$$\rho_z(j) = \frac{1}{M-j}\sum_{i=1}^{M-j} z_i z_{i+j}, \quad j = 0, \ldots, M-1. \tag{C.4}$$

Note that there are $M-j$ values contributing to $\rho_G(j)$, and for j close to M the statistical errors are large. For instance, for $j = M-1$, there is only one contribution to $\rho_z(j)$, namely z_1z_M. In addition, when M is large, the direct calculation of $\rho_z(j)$ using (C.4) could be slow, as it takes of the order of M^2 operations. We now show that it is possible to greatly reduce the time needed to compute a correlation function using the discrete Fourier transform.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

As explained in Appendix G, the discrete Fourier transform $\hat{x} = \mathcal{F}_D[x]$ of an arbitrary set of numbers $(x_1, \dots, x_n)$ is defined as[1)]

$$\hat{x}_k = \sum_{j=0}^{n-1} e^{\frac{2\pi i}{n} jk} x_{j+1}, \qquad k = 0, 1, \dots, n-1 \tag{C.5}$$

and the inverse relation, indicated as $x = \mathcal{F}_D^{-1}[\hat{x}]$

$$x_{j+1} = \frac{1}{n} \sum_{k=0}^{n-1} e^{-\frac{2\pi i}{n} jk} \hat{x}_k, \qquad j = 0, \dots, n-1. \tag{C.6}$$

With the help of (G.10), it is possible to prove the identity

$$\sum_{i=1}^{n} x_i x_{i+j} = \frac{1}{n} \sum_{k=0}^{n-1} \mathrm{e}^{-\frac{2\pi i}{n} jk} |\hat{x}_k|^2 \tag{C.7}$$

which, given (C.4), seems to imply that $\mathcal{F}_D^{-1}[|\hat{x}|^2]$, the inverse discrete Fourier transform of the series $(|\hat{x}_0|^2, \dots, |\hat{x}_{n-1}|^2)$, is related in some way to the correlation function $\rho_x(j)$ of the series x. This is true, but one has to be careful. The point is that, for the previous formula (C.7) to be exact, one has to assume periodic boundary conditions, that is, whenever x_{i+j} appears in the left-hand side of this formula with $i + j > n$, it has to be understood as x_{i+j-n}. This property, namely $x_j = x_{j-n}$ for $j > n$, which derives directly from the extension of (C.6) to values $j > n$, is not present in the original series as it runs from x_1 to x_n, and x_j does not even exist for $j > n$. Note that, in (C.4), for example, one never uses z_i with $i \geq M$. With a small trick, we can relate exactly the correlation function $\rho_z(j)$ to an inverse discrete Fourier transform. The trick is to introduce a new series $x = (x_1, \dots, x_n)$ of length $n = 2M$, which is twice the length of the original z series. The new series is defined as

$$x_i = \begin{cases} z_i, & i = 1, \dots, M, \\ 0, & i = M+1, \dots, 2M. \end{cases} \tag{C.8}$$

It is a straightforward consequence of this definition that the sum needed in (C.4) for the calculation of $\rho_z(j)$ can be written as

$$\sum_{i=1}^{M-j} z_i z_{i+j} = \sum_{i=1}^{n} x_i x_{i+j}, \quad j = 0, 1, \dots, M-1 \tag{C.9}$$

with periodic boundary conditions now assumed in the sum of the right-hand side.[2)] Finally, using (C.4), (C.8), and the identity (C.7) with $n = 2M$, we get

$$\rho_z(j) = \frac{1}{M-j} \left[\frac{1}{n} \sum_{k=0}^{n-1} \mathrm{e}^{-\frac{2\pi i}{n} jk} |\hat{x}_k|^2 \right], \quad j = 0, 1, \dots, M-1 \tag{C.10}$$

showing that it is possible to obtain, after multiplication by the factor $\frac{1}{M-j}$, the correlation function $\rho_z(j)$ from the inverse discrete Fourier series of $|\hat{x}_k|^2$, with $\hat{x}_k$

1) For the difference in notation with respect to Appendix G we refer to the footnote B.2).
2) The reader might find it useful to check by him/herself (C.7) and (C.9).

being the direct discrete Fourier series of x_i. The big advantage of this procedure is that the use of fast Fourier routines allows one to implement this algorithm in a time that scales as $M \log M$, which is much smaller for large M than the time of order M^2 needed by the direct algorithm.

Once we have computed $\rho_G(j)$, we can compute the correlation time τ_G using (4.23). Just a final word of warning: if you plot the resulting correlation function, it will typically behave smoothly only for not too large values of j and it will have wild oscillations for large j. The reason is that the larger the value of j, the fewer the values that contribute to its estimator, which increases the error greatly. As a consequence, when computing τ_G using (4.23), do not extend the upper sum all the way to $M-1$. Restrict the sum to those values of j for which the correlation function, having decayed to a value near zero, is still well behaved. In fact, if all you want is to have an estimate of the correlation time τ_G in order to determine faithfully the error of your estimator, it is enough to know the characteristic decay time of the correlation function.[3)]

Here is a program listing that computes the correlation function of a given series. Remember to read the instructions of the fast Fourier transform routines to find out whether you need to correct for any factors of M. In this subroutine, the correlation function $\rho_z(i)$ overwrites the input value x_i. It returns also the mean `xm` and the root-mean-square `x2`.

```
subroutine correla(x,m,xm,x2)
implicit double precision (a-h,o-z)
dimension x(0:m-1)
double complex z(0:2*m-1)

xm=sum(x)/m
x2=sqrt(sum(x**2)/m-xm*xm)
x=(x-xm)/x2
z(0:m-1)=dcmplx(x,0.0d0)
z(m:2*m-1)=dcmplx(0.0d0,0.0d0)
call fft1d(z,2*m,1)
z=dcmplx(abs(z)**2,0.0d0)
call fft1d(z,2*m,-1)
do j=0,m-1
  x(j)=real(z(j))/(m-j)
enddo

end subroutine correla
```

3) There is another deeper reason for not summing all the way up to $j = M-1$ to obtain the correlation time, as it can be shown that this procedure leads to $\tau_G = -1/2$ and hence a zero error in the estimator. The reason is simple to understand: we are computing the error of an estimator using the own estimator as the true value. In other words, we are taking formula (4.15) with $I = \frac{1}{N}\sum_{k=1}^{M} G_k$ and hence the error is 0.

Exercises

1) Prove (C.7) and (C.9).

Appendix D
Collective Algorithms for Spin Systems

We have argued in our general exposition of the dynamical methods to generate representative configurations according to a given pdf that it is important, in order to keep a reasonable acceptance probability, that the proposed new configuration should not be very different from the old one. In fact, in our implementation of the Metropolis algorithm in many-variable systems, we chose to change only a few variables, maybe just only one, at a time. For example, in the Ising model, one spin s_i was selected and the proposed change was $s_i \to -s_i$. Since every configuration is very much correlated with the previous one (only a few variables might be different), the correlation time τ of the algorithm is very high. In fact, we showed in (5.121) that it diverges with some power L^z of the system side, the *critical slowing down*. Is it possible to make a big change in the configuration and still keep a small correlation time? The hybrid Monte Carlo method discussed in Chapter 10 aims precisely at this for continuous systems. However, despite its many advantages, it does not succeed in avoiding the critical slowing down problem.

For spin systems,[1)] very clever algorithms have managed to implement a proposal that, although it implies a big change with respect to the current configuration, it is always accepted (as in heat-bath methods), leading to a drastic reduction of critical slowing down. The first collective update algorithm was proposed by Swendsen and Wang [70] based on the Fortuin and Kasteleyn mapping of the Ising and Potts models onto a percolation model. The idea is to connect neighboring states with the same value of the spin using a carefully chosen probability p. For the Ising model (to which we will restrict our discussion in this Appendix), it is $p = 1 - e^{-2J/kT}$ (J is the coupling constant of the Ising Hamiltonian, see (5.45) with $H = 0$). Once the connections have been created, the system decomposes into a series of ''clusters'' (sets of connected spins). A new value of the spin variable is assigned randomly to each cluster and the same value to each site of the cluster. This algorithm represented a landmark in the field of simulations of the Ising and Potts systems, as it was shown to greatly reduce the exponent z of the divergence of the correlation time τ at criticality.

A more efficient algorithm was soon developed by Niedermayer [71] and, independently, by Wolff [72]. The idea of both algorithms is to build a cluster of

1) Representative examples are Ising and Potts models, but the methods we will explain can be made more general.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

connected sites and then "flip" (change the sign of) all spins in that cluster. While in the Niedermayer algorithm a cluster can contain spins of arbitrary sign, the Wolff one limits the clusters to containing only one common value of the spin. Although the Wolff algorithm can be considered a particular case of the Niedermayer algorithm, it turns out that the former is more efficient in the sense that it can reduce drastically the correlation time near the critical point [73].

To be more precise, the Niedermayer algorithm proceeds by first selecting one site, i, randomly. The variable at this site, s_i, can be either +1 or −1. This site will be the seed of the cluster. We run over its nearest neighbors i_μ and add them to the cluster with a probability p_a if the value of the spin s_{i_μ} is equal to s_i, or with a probability p_b if s_{i_μ} has the opposite sign than s_i. The process continues recursively until no more additions are produced. Then, the whole cluster of spins is flipped. The Wolff algorithm is a particular case with $p_b = 0$.

Let us explain the simpler Wolff algorithm using a particular example of the process of construction of the clusters. Look at the configuration marked X in Figure D.1. Imagine that the algorithm leads to the creation of the cluster C formed by five particles, which we label by (1, 2, 3, 4, 5). In configuration X, this cluster has a value of spin −1, while in configuration Y the same cluster C appears with a value +1. What is the probability of having created cluster C in X? that is, the probability of proposing a change $X \to Y$ in which all spins of cluster C have been flipped. In the language of Chapter 4, we are asking for the proposal probability $g(Y|X)$. At the bottom of the figure, we have detailed *t*wo out of the many possible ways of creating this cluster. We call **A** and **B** these two particular sequence of additions leading to the formation of cluster (1, 2, 3, 4, 5) in X.

In sequence **A**, we have first selected site 1 (this has happened with probability $1/N$), then we look at the neighbors of 1 which are in the same state, −1 in this case. They are 2, 5, and 14. It happens that 2 and 5 are added to the cluster (a success in each case with probability p_a) but 14 is not (probability $1 - p_a$). Since 2 and 5 have been added, we repeat the process beginning with these two new sites. First, check the same-state neighbors of site 2 that are not yet part of the cluster (for short, we will name these the "candidate sites"); the only one is 3, which happens to be rejected (probability $1 - p_a$). Then check the candidate sites of 5; they are 3, 11, and 12. It happens that 3 is now added (probability p_a) and 11 and 12 are not (each one with probability $1 - p_a$). Search for the candidate sites of 3; the only one is site 4 that is accepted (probability p_a). The final step is to check for the set of candidate sites of 4, which is limited to 9 and this happens to be rejected (probability $1 - p_a$). Counting all factors, the probability of creating the cluster C is $\frac{1}{N} p_a^4 (1 - p_a)^5$.

In sequence **B**, we first select site 2 (probability $1/N$), the candidate sites are 1 (accepted) and 3 (accepted). The candidate sites of 1 are 14 (rejected) and 5 (accepted); the candidate sites of 5 are 11 (rejected) and 12 (rejected). The only candidate site of 3 is 4 (accepted); the only candidate site of 4 is 9 (rejected). Counting the accepted and rejected sites, the probability of creating the cluster C is $\frac{1}{N} p_a^4 (1 - p_a)^4$.

Each path leading to the selection of cluster C in configuration X has a probability $\frac{1}{N} p_a^{n_a} (1 - p_a)^{n_i + n_X}$, with n_a being the number of accepted additions, while the

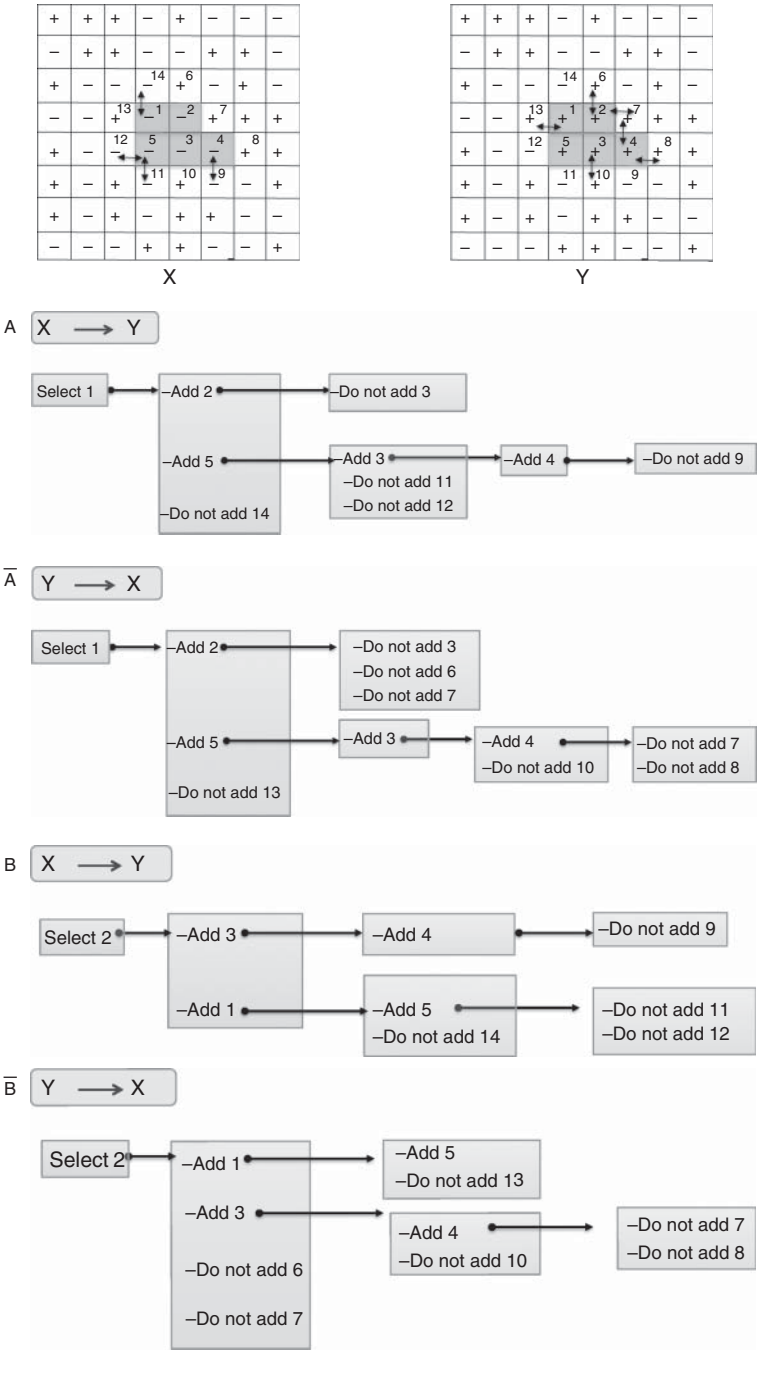


Figure D.1 Example of the cluster construction sequences A, $\overline{A}$, B, and $\overline{B}$ used to illustrate the Wolff method.

number of rejections has been split as $n_i + n_X$, with n_i being the number of rejections of sites that, in the end, form part of the cluster, while n_X is the number of rejections of sites that do not form part of the cluster. n_X is the number of links between spins of the same sign that happen at the border of the cluster. $n_X = 4$ for this particular example. They are indicated by arrows in configuration X of Figure D.1. The proposal probability of Y given X is then

$$g(Y|X) = \sum_{\text{Paths leading to C}} \frac{1}{N} p_a^{n_a} (1-p_a)^{n_i+n_X} \tag{D.1}$$

$$= \frac{1}{N} p_a^{n_a} (1-p_a)^{n_X} \sum_{\text{Paths leading to C}} (1-p_a)^{n_i} \tag{D.2}$$

as n_a and n_X are common to all paths.

We note that each of the previous sequences, **A**, **B**, and so on, has an inverse sequence named $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, and so on, which leads to the selection of cluster C in configuration Y. This inverse path is created by accepting the same additions that were accepted in the direct path and rejecting all others. For instance, the inverse path $\overline{\mathbf{A}}$ leading to the selection of cluster C in Y follows the steps: Select site 1 (probability $1/N$); the same state, +1, neighbors not belonging to the cluster (the candidate sites) are now 2, 5, and 13; we add 2 and 5 (because they were accepted in path **A**) and reject 13; the candidate sites of 2 are 3, 6, and 7, which are all rejected; the only candidate site of 5 is 3 and this is added; the candidate sites for 3 are 4 (accepted) and 10 (rejected); finally, the candidate sites of 4 are 7 and 8 and we reject both. Counting accepted and rejected sites, this path has a probability of being chosen of $\frac{1}{N} p_a^4 (1-p_a)^7$.

The inverse sequence $\overline{\mathbf{B}}$ is the following: Select site 2 (probability $1/N$); the candidate sites are 1 (accepted), 3 (accepted), 6 (rejected), 7 (rejected); the candidate sites of 1 are 5 (accepted) and 13 (rejected); the candidate sites of 3 are 4 (accepted) and 10 (rejected); the candidate sites of 4 are 7 and 8 (both rejected). Counting the accepted and rejected sites, the probability of creating the cluster C using this path is $\frac{1}{N} p_a^4 (1-p_a)^6$.

The proposal of the reverse path to select configuration Y has a probability $\frac{1}{N} p_a^{n_a} (1-p_a)^{n_i+n_Y}$, where n_a and n_i are the same as in the selection of configuration X, and n_Y is the number of links between spins of the same sign that happen at the border of the cluster, as indicated by arrows in configuration Y of Figure D.1, $n_Y = 6$ in our example. So

$$g(X|Y) = \frac{1}{N} p_a^{n_a} (1-p_a)^{n_Y} \sum_{\text{Paths leading to C}} (1-p_a)^{n_i}. \tag{D.3}$$

Hence, the two probabilities satisfy

$$g(Y|X)(1-p_a)^{n_Y} = g(X|Y)(1-p_a)^{n_X}. \tag{D.4}$$

Note that $n_X + n_Y$ is equal to the total number of links at the border of cluster C. The difference in energy between configurations X and Y is due solely to the links at the border of the cluster. Every one of the n_X links in configuration X, when flipped, increases the energy in $2J$, while the other n_Y links decrease the energy by the same amount. The difference in energies is then $\mathcal{H}(Y) = \mathcal{H}(X) + 2J(n_X - n_Y)$.

If we replace this and (D.4) in the detailed balance condition

$$g(Y|X)h(Y|X)e^{-\beta\mathcal{H}(X)} = g(X|Y)h(X|Y)e^{-\beta\mathcal{H}(Y)} \tag{D.5}$$

we obtain

$$\frac{h(Y|X)}{h(X|Y)} = \left[e^{2\beta J}(1-p_a)\right]^{n_Y - n_X}. \tag{D.6}$$

So far, the addition probability p_a is arbitrary. We notice that, if we choose $e^{2\beta J}(1-p_a) = 1$ or

$$p_a = 1 - e^{-2\beta J} \tag{D.7}$$

then the ratio between the acceptance probabilities becomes $\frac{h(Y|X)}{h(X|Y)} = 1$. Furthermore, this relation is satisfied by the simple choice $h(Y|X) = h(X|Y) = 1$. We have achieved a heat-bath type rejection-free algorithm (every proposal is accepted) while the number of changed variables might be large. The detailed analysis by Wolff and others shows that, indeed, the changes from one configuration to another are so radical near the critical point that the algorithm is basically free of the critical slowing down problem. The correlation time is shown to grow only as the logarithm of the number of spin variables, instead of a power law, as in the Metropolis and similar algorithms. Similar efficient algorithms have been also proposed to study Potts, XY, and hard-sphere models, amongst others.

We now give the basic commands of the programming of such an algorithm for the square lattice where every site has four nearest neighbors. Besides all definitions of vectors, neighbors, and so on, common to the program in Section 5.4, the updating part is the following: First select one site randomly and add it to the cluster, then run over its neighbors to check whether they should be accepted or not. This is done by the routine `addtocluster`. This routine is recursive: if a neighbor is selected to be added to the cluster, then it runs over its neighbors and calls again the same routine. Despite the apparent difficulty of the Wolff algorithm, the routine is extremely simple, which has certainly contributed to its widespread use.

```
 i=i_ran(N)
 call addtocluster(i,s,neigh,N,pa)
```

```
recursive subroutine addtocluster(i,s,neigh,N,pa)
dimension neigh(4,N)
integer s(N)
s(i)=-s(i) ! Flip the particle just added to the cluster
do k=1,4
 j=neigh(k,i)
 if(s(j) == -s(i)) then
 if (ran_u() < pa) call addtocluster(j,s,neigh,N,pa)
 endif
enddo
end
```

Note that, as the first thing the routine does is to flip the value of the spin variable `s(i)` that has been accepted to the cluster, the check that new spins can be added requires the comparison of `s(j)` with the old value of `s(i)` which is `-s(i)`.

From a practical point of view, the Swenden--Wang algorithm is similar to the Wolff one but many clusters are created at once and flipped with probability 1/2.

It would be a good exercise for the reader to redo the previous reasoning in the case of the Niedermayer algorithm. The new ingredient is that now it is also allowed to add, with probability p_b, a nearest neighbor spin with a different sign. The detailed balance condition reads now

$$\frac{h(Y|X)}{h(X|Y)} = \left[\frac{1-p_a}{1-p_b} e^{2\beta J}\right]^{n_Y - n_X} . \tag{D.8}$$

An acceptance probability equal to 1 requires $\frac{1-p_a}{1-p_b} e^{2\beta J} = 1$. Wolff algorithm takes precisely $p_b = 0$. This turns out to be the optimal choice [73, 74].

Appendix E
Histogram Extrapolation

In many occasions, we need to compute an average value and study its variation with a parameter. This is the case, for example, of the calculation of the dependence of the average magnetization as a function of the temperature, $\mathtt{m}(T)$. For the particular case of the Ising model, this was obtained as $\mathtt{m}(T) = \langle m \rangle$ with the function

$$m(s_1, \ldots, s_N) = \left| \frac{1}{N} \sum_{i=1}^{N} s_i \right| \tag{E.1}$$

and averages performed with respect to the N-dimensional pdf given by (5.54).

For a given value of the temperature T, we can use any of the sampling methods explained to obtain a good estimator of $\mathtt{m}(T)$ and its error. In order to study the variation of the magnetization with temperature, it seems that we must repeat the sampling process for different values of the temperature. However, there is a powerful technique analyzed in detail by Ferrenberg and Swendsen [75], which allows one to obtain the whole variation of $\mathtt{m}(T)$ over a range of temperatures by performing, essentially, only one sampling process at a single value of the temperature. Let us now explain how this miracle is possible.

The average values as a function of temperature T of any function $G(X)$ of the relevant system degrees of freedom $X = (x_1, \ldots, x_N)$ are found as

$$\langle G \rangle_T = \frac{\int dX\, G(X) e^{-\beta \mathcal{H}(X)}}{\int dX\, e^{-\beta \mathcal{H}(X)}}, \quad \beta = 1/kT \tag{E.2}$$

which, of course, is nothing but the average value of $G(X)$ with respect to the pdf:

$$f_{\hat{\mathbf{X}}}(X; T) = \frac{e^{-\beta \mathcal{H}(X)}}{\mathcal{Z}(T)}, \qquad \mathcal{Z}(T) = \int dX\, e^{-\beta \mathcal{H}(X)}, \tag{E.3}$$

and, in order to stress the dependence with the temperature, we use the notation $\langle G \rangle_T$. Now, it is a matter of simple algebra to prove that for arbitrary T, T' it is

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

$$
\begin{aligned}
\langle G\rangle_{T'} &= \frac{\int dX\, G(X) e^{-\beta' \mathcal{H}(X)}}{\int dX\, e^{-\beta' \mathcal{H}(X)}} \\
&= \frac{\int dX\, G(X) e^{-(\beta'-\beta)\mathcal{H}(X)} e^{-\beta \mathcal{H}(X)}}{\int dX\, e^{-\beta \mathcal{H}(X)}} \frac{\int dX\, e^{-\beta \mathcal{H}(X)}}{\int dX\, e^{-\beta' \mathcal{H}(X)}} \\
&= \frac{\int dX\, G(X) e^{-(\beta'-\beta)\mathcal{H}(X)} e^{-\beta \mathcal{H}(X)}}{\int dX\, e^{-\beta \mathcal{H}(X)}} \Big/ \frac{\int dX\, e^{-(\beta'-\beta)\mathcal{H}(X)} e^{-\beta \mathcal{H}(X)}}{\int dX\, e^{-\beta \mathcal{H}(X)}} \\
&= \int dX\, G(X) e^{-(\beta'-\beta)\mathcal{H}(X)} f_{\hat{\mathbf{X}}}(x;\beta) \Big/ \int dX\, e^{-(\beta'-\beta)\mathcal{H}(X)} f_{\hat{\mathbf{X}}}(x;\beta) \\
&= \frac{\left\langle G(X) e^{-(\beta'-\beta)\mathcal{H}(X)} \right\rangle_T}{\left\langle e^{-(\beta'-\beta)\mathcal{H}(X)} \right\rangle_T}. \qquad \text{(E.4)}
\end{aligned}
$$

The averages of the last line of the above formula are with respect to the distribution $f_{\hat{\mathbf{X}}}(X;T)$ but, as shown, that particular ratio gives us the average value of $G(X)$ with respect to $f_{\hat{\mathbf{X}}}(X;T')$. We can then use the generation of random numbers distributed according to $f_{\hat{\mathbf{X}}}(X;T)$ to compute averages with respect to $f_{\hat{\mathbf{X}}}(X;T')$ for a different value $T' \neq T$. This is the miracle. Remember that in N-dimensional calculations, the most painful part is the generation of the random numbers distributed according to a complicated pdf. This result tells us that we need to do this complicated part only once for a particular value of temperature, and then we can extrapolate, that is, obtain averages for any other value of the temperature.

Now the bad news: it is not true *in practice* that we can obtain averages *for any other value* of the temperature. There is a range of values of the temperature difference $|T - T'|$ for which the extrapolation can proceed with small statistical errors. Before analyzing this, let us first see how we would implement (E.4). We select a value of temperature T and generate M configurations $X^{(k)}, k = 1, \ldots, M$ distributed according to $f_{\hat{\mathbf{X}}}(X;\beta) = \dfrac{e^{-\beta \mathcal{H}(X)}}{\mathcal{Z}(\beta)}$ using our favorite method. For each configuration $X^{(k)}$, we compute the value of the Hamiltonian $E^{(k)} = \mathcal{H}(X^{(k)})$ and $G^{(k)} = G(X^{(k)})$. With these values of $E^{(k)}$ and $G^{(k)}$, we construct a histogram $H_T(E, G)$ (hence the name of histogram extrapolation). If (as in the case of the Ising model, for instance) the possible values $E^{(k)}$ and $G^{(k)}$ form a discrete set $E_0, E_1, \ldots, E_{M_E}$, $G_0, G_1, \ldots, G_{M_G}$, then the histogram value $H_T(E_i, G_j)$ counts the number of times, $\Omega(E_i, G_j)$, the particular pair (E_i, G_j) appears divided by the total number of configurations M. In other cases, we introduce appropriate bin sizes ΔE, ΔG, set $E_i = E_0 + i\Delta E$, $G_j = G_0 + j\Delta G$ (with suitable minimum values E_0, G_0), and define $H_T(E_i, G_j) = \dfrac{\Omega(E_i, G_j)}{M \Delta E \Delta G}$ being now $\Omega(E_i, G_j)$, the number of configurations that yield a value of $\mathcal{H}$ in the interval (E_i, E_{i+1}) and G in the interval (G_j, G_{j+1}). For the sake of concreteness, we adopt from now on the notation of the discrete case. Once the histogram $H_T(E_i, G_j)$ has been constructed, the averages needed in (E.4) are computed as

$$
\left\langle G(X) e^{-(\beta'-\beta)\mathcal{H}(X)} \right\rangle_T = \sum_{i,j} H_T(E_i, G_j) G_j e^{-(\beta'-\beta)E_i} \qquad \text{(E.5)}
$$

$$\left\langle e^{-(\beta'-\beta)\mathcal{H}(X)} \right\rangle_T = \sum_{i,j} H_T(E_i, G_j) e^{-(\beta'-\beta)E_i}. \tag{E.6}$$

In fact, in the right-hand-side of this equation we can use just the histogram of the energy $H_T(E_i) = \sum_j H_T(E_i, G_j)$

$$\left\langle e^{-(\beta'-\beta)\mathcal{H}(X)} \right\rangle_T = \sum_{i} H_T(E_i) e^{-(\beta'-\beta)E_i}. \tag{E.7}$$

The histograms $H_T(E_i, G_j)$ might occupy a large amount of computer memory. Just consider the Ising model for which the distance between the minimum and maximum value of energy scale is $M_E = O(N)$. If, for example, $G(X)$ is the magnetization, then similarly $M_G = O(N)$, hence the histogram $H_T(E_i, G_j)$ has of the order of N^2 entries, a possibly large number. It is possible to avoid the use of histograms if we set beforehand the values $T'_1, T'_2, \ldots$ where we want to use the extrapolation formula (E.4). We simply add a contribution to the average value every time we generate a configuration $X^{(k)}$, using

$$\left\langle G(X) e^{-(\beta'_i-\beta)\mathcal{H}(X)} \right\rangle_T \approx \frac{1}{M} \sum_{k=1}^{M} G(X^{(k)}) e^{-(\beta'_i-\beta)\mathcal{H}(X^{(k)})}, \tag{E.8}$$

$$\left\langle e^{-(\beta'_i-\beta)\mathcal{H}(X)} \right\rangle_T \approx \frac{1}{M} \sum_{k=1}^{M} e^{-(\beta'_i-\beta)\mathcal{H}(X^{(k)})} \tag{E.9}$$

looping over the desired values of β'_i.

Although the above extrapolation formula (E.4) is exact, its practical validity is limited to values of β' that are not too distant from β. To understand why, let us look in detail at the denominator of this expression (a similar argument holds for the numerator). When we use (E.7) (or equivalently (E.9)) to estimate this average, we use a histogram $H_T(E_i)$ that has been constructed with values of the random variable $X^{(k)}$ distributed with the pdf $f_{\hat{\mathbf{X}}}(X;\beta)$, which generally is a function that, as discussed in previous chapters, has a pronounced maximum at an unknown location. This means that $H_T(E_i)$ has a large peak at some energy value $U(T)$, but—and this is the key point—the function $H_T(E_i)e^{-\Delta\beta' E_i}$, with $\Delta\beta' = \beta' - \beta$, has a peak at another value $U(T')$. If $U(T')$ differs significantly from $U(T)$, the histogram values of $H_T(E)$ around $E = U(T')$ will have a large statistical error (they are part of the tail of the histogram of $H_T(E)$) and hence the whole average has a large error, see Figure E.1. A rule of thumb is that the distance $|U(T) - U(T')| = |\Delta U|$ has to be of the order of the width, which is the standard deviation $\sigma_T[E]$ of the distribution of the histogram $H_T(E)$. But Einstein's relation (5.51)

$$\sigma_T^2[E] = kT^2 C_V(\beta) \tag{E.10}$$

relates the width $\sigma_T[E]$ to the specific heat $C_V(T) = \dfrac{dU(T)}{dT}$ or, replacing the derivative by a ratio of finite differences, $C_V(T) \approx \dfrac{\Delta U}{\Delta T}$, from where $|\Delta T| = \dfrac{|\Delta U|}{C_V(T)}$.

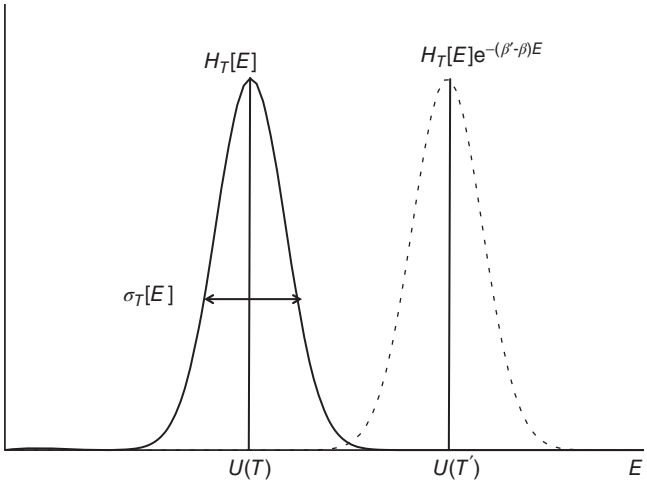


Figure E.1 Relation between the original $H_{T'}(E)$ and the extrapolated histogram $H_{T'}e^{-(\beta'-\beta)E}$.

If we now replace $|\Delta U| \sim \sigma_T[E] = \sqrt{kT^2 C_V(T)}$, we get

$$\frac{|\Delta T|}{T} \sim \sqrt{\frac{k}{C_V(T)}} \tag{E.11}$$

as the range of values $|\Delta T|$ for which the histogram extrapolation can be carried out safely. The not-so-good news is that the specific heat C_V is an extensive quantity that generally scales linearly with the number of degrees of freedom, $C_V \sim N$. This means that the range of values of T for which a good extrapolation can be obtained scales as $|\Delta T| \sim N^{-1/2}$ or $|\Delta T| \sim L^{-d/2}$ using the system length defined by $N \sim L^d$, for spatial dimension d. Most simulations care about the behavior near the critical point, T_c. The situation here seems to be worse, as the specific heat per particle at the critical point $C_V(T_c)/N$ diverges. However, as discussed at the end of Chapter 5, this divergence for a system of characteristic linear size L scales as $C_V(T_c)/N \sim L^{\alpha/\nu}$ or $C_V(T_c) \sim L^{2/\nu}$ using the hyperscaling relation $2 - \alpha = d\nu$. This means that at the critical point the range of values of T for which extrapolation is meaningful scales instead as $|\Delta T| \sim L^{-1/\nu}$.

In a further extension [76], Ferrenberg and Swendsen have shown how to use information coming from a set of simulations at different selected values of the temperature in order to extrapolate to a much larger extent. These extrapolations techniques can be used also for many systems of interest in statistical mechanics and are nowadays a standard simulation tool.

Appendix F
Multicanonical Simulations

So far, our sampling method has allowed us to obtain good estimates for some quantities of interest, such as the magnetization m, the internal energy U, the specific heat C_V, and so on. All these quantities can be expressed as averages with respect to the Boltzmann factor. However, the entropy cannot be written as an average and it cannot be obtained readily with the methods explained so far. This is where the multicanonical simulations appear. The general methodology is to sample distributions other than the Boltzmann factor and then, besides allowing for the calculation of the entropy, can be used to speed up or to extend the usual canonical simulations. There is a vast bibliography in this topic and we cannot even aim to summarize in a comprehensive way all recent developments in this area. Some topics that we cannot cover but are somewhat related to multicanonical simulations include the field of simulated tempering, where the temperature becomes a dynamical variable changing during the updates [77]; parallel tempering, also known under the names of replica exchange, exchange Monte Carlo, and multiple Markov chain Monte Carlo [78–80]; umbrella sampling [81]; broad histograms [82]; transition matrix Monte Carlo [83], and so on, for which we refer the reader to more specialized bibliography.

Imagine we perform a Monte Carlo sampling and generate representative configurations $X^{(1)}, \ldots, X^{(M)}$ distributed according to the Boltzmann pdf $f_{\hat{\mathbf{X}}}(X) = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}(X)}$. In the process, we measure the energy of these configurations $E^{(i)} = \mathcal{H}(X^{(i)})$. The values $E^{(i)}$ are distributed according to the pdf $f_{\mathcal{H}}(E) = \mathcal{Z}^{-1} \Omega(E) e^{-\beta E}$, with $\Omega(E)$ being the number of configurations that have energy E.[1)] Using the fact that the density of states is related to the entropy by Boltzmann's relation $S(E) = \log \Omega(E)$,[2)] we can write $f_{\mathcal{H}}(E) = \mathcal{Z}^{-1} e^{-\beta E + S(E)}$. With our simulation, we can produce a histogram $H_T(E)$ of the energy values. This histogram is an approximation (within the unavoidable statistical errors) to the true pdf $f_{\mathcal{H}}(E)$. We can use this histogram $H_T(E) \approx \mathcal{Z}^{-1} e^{-\beta E + S(E)}$ to estimate the entropy as

$$S(E) \approx S_0 + \log H_T(E) + \beta E \tag{F.1}$$

1) We use, for simplicity, the case in which there is a numerable set of possible energies: $E_0, E_1, \ldots$. If the energy can adopt any real value, $\Omega(E)dE$ is the number of configurations with energy in the interval $(E, E + dE)$. Although in the main text we use the discrete notation, we might refer to $\Omega(E)$ as the "density" of states.

2) More precisely, $S(E) = k \log \Omega(E)$, with k the Boltzmann's constant. We adopt henceforth $k = 1$.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

with $S_0 = \log \mathcal{Z}$, a constant independent of E. This is a remarkable formula, as it gives us the full functional dependence (except an irrelevant additive constant) of the entropy with respect to the energy, $S(E)$, using a single simulation at inverse temperature β. Alas! as the reader might guess by now, since the situation is similar to the problems encountered by the Ferrenberg–Swendsen extrapolation scheme discussed in Appendix E, the practical validity of this formula is limited to an interval of energies for which the histogram $H_T(E)$ can be determined with enough precision. The maximum of $H_T(E)$ occurs at a value $E = U(T)$ corresponding to the maximum of the argument of the exponential $-\beta E + S(E)$, or

$$\left.\frac{dS(E)}{dE}\right|_{E=U(T)} = \beta \tag{F.2}$$

and the width of $H_T(E)$ can be related to its variance $\sigma_T^2[E]$, as given by

$$\sigma_T^{-2}[E] = -\left.\frac{d^2 S(E)}{dE^2}\right|_{E=U(T)} \tag{F.3}$$

or $\sigma_T^2[E] = \beta^{-2} C_V(\beta)$, Einstein's relation, with C_V being the specific heat. As $C_V = O(N)$, it turns out that $\sigma_T = O(N^{1/2})$.[3] As the typical span of values of E is of the order N, the relative range of values covered of E which are well represented in a canonical simulation at inverse temperature β is of the order $N^{-1/2}$, which is not very large and limiting considerably the practical validity of (F.1).

Can we do better than this to determine the entropy $S(E)$ for a larger range of energy values? Yes, we can. All we need is to use another sampling function instead of $f_{\hat{\mathbf{X}}}(X) = \mathcal{Z}^{-1} e^{-\beta \mathcal{H}(X)}$. Let us consider the general pdf

$$f_{\hat{\mathbf{X}}}^{\omega}(X) = C e^{-\omega(X)} \tag{F.4}$$

where the function $\omega(X)$ depends on the configuration only through the value of the Hamiltonian, $\omega(X) = \omega(\mathcal{H}(X))$, and C is the normalization constant. If we now sample this distribution, generate $X^{(1)}, X^{(2)}, \ldots, X^{(M)}$ configurations and compute a histogram $H_\omega(E)$ of the energies of these configurations $E^{(i)} = \mathcal{H}(X^{(i)})$, $i = 1, 2, \ldots, M$, then $H_\omega(E)$ will be an approximation to the pdf $f_\omega(E) = C\Omega(E) e^{-\omega(E)} = C e^{-\omega(E)+S(E)}$. We can then approximate the entropy from the simulation

$$S(E) \approx S_0 + \omega(E) + \log(H_\omega(E)) \tag{F.5}$$

with, again, S_0 a constant. Which function $\omega(E)$ can give us the largest domain of practical validity of this formula? After a moment of thought, it naturally appears that the optimal choice is $\omega(E) = S(E) = \log \Omega(E)$, since in this case the distribution $f_\omega(E) = C e^{-\omega(E)+S(E)} = C$ is a constant for all values of E and therefore there are no preferred values of E. The histogram $H_\omega(E)$ is perfectly flat and the practical validity of (F.5) extends to all values of E. However, it is clear that this optimal choice $\omega(E) = S(E)$ is yet useless, as $S(E)$ is precisely the function we want to determine! Nevertheless, a series of techniques have been developed in order to

3) Or $\sigma_T = O(N^{1/d_\nu})$ at the critical point, following the argument explained for the Ferrenberg–Swendsen extrapolation method.

compute, using these and other ideas, the entropy $S(E)$ or, equivalently, the density of states $\Omega(E) = e^{S(E)}$. All these methods are approximate and yield a better or worse approximation $S_{\text{app}}(E)$ to the true entropy $S(E)$.

Before explaining these methods, let us explain what would be the advantage of a Monte Carlo simulation using the pdf $f_{\hat{\mathbf{X}}}^{S_{\text{app}}}(X) = Ce^{-S_{\text{app}}(X)}$ instead of the standard $f_{\hat{\mathbf{X}}}(X) = \mathcal{Z}^{-1}e^{-\beta\mathcal{H}(X)}$. First of all, a simulation using $f_{\hat{\mathbf{X}}}^{S_{\text{app}}}(X)$ is called *multicanonical*.[4] A Monte Carlo multicanonical simulation will allow us to obtain averages of any function $G(X)$ as

$$\langle G(X)\rangle_{\text{multi}} \equiv \frac{\int dXG(X)e^{-S_{\text{app}}(X)}}{\int dXe^{-S_{\text{app}}(X)}}. \tag{F.6}$$

We can relate in a simple way the desired averages with respect to the canonical distribution at temperature T to the ones obtained in a multicanonical simulation:

$$\langle G(X)\rangle_T = \frac{\int dXG(X)e^{-\beta\mathcal{H}(X)}}{\int dXe^{-\beta\mathcal{H}(X)}} \tag{F.7}$$

$$= \frac{\int dX\left[G(X)e^{-\beta\mathcal{H}(X)+S_{\text{app}}(X)}\right]e^{-S_{\text{app}}(X)}}{\int dX\left[G(X)e^{-\beta\mathcal{H}(X)+S_{\text{app}}(X)}\right]e^{-S_{\text{app}}(X)}} \tag{F.8}$$

$$= \frac{\langle G(X)e^{-\beta\mathcal{H}(X)+S_{\text{app}}(X)}\rangle_{\text{multi}}}{\langle e^{-\beta\mathcal{H}(X)+S_{\text{app}}(X)}\rangle_{\text{multi}}} \tag{F.9}$$

$$\approx \frac{\sum_{i=1}^{M} G(X^{(i)})e^{-\beta\mathcal{H}(X^{(i)})+S_{\text{app}}(X^{(i)})}}{\sum_{i=1}^{M} e^{-\beta\mathcal{H}(X^{(i)})+S_{\text{app}}(X^{(i)})}} \tag{F.10}$$

where in the last line we have used the Monte Carlo estimators using the configurations $X^{(i)}$ obtained in the multicanonical simulation. And now it is apparent that averages with respect to any value of T can be computed using a single multicanonical distribution. And this time "any value of T" is true (if we use a good approximation for $S_{\text{app}}(X)$), a big improvement indeed. Note that, irrespective of whether $S_{\text{app}}(X)$ is a good or bad approximation to the true (unknown) entropy $S(X)$, (F.10) is an estimator of the thermal average (F.7). It is only the practical validity of the estimator (F.10) for a wide range of values of β which is optimized if $S_{\text{app}}(X)$ is very close to $S(X)$.

So, how do we obtain an approximation S_{app} for the entropy S? The first approach to this problem was given by Bhanot *et al.* [84], which used a series of microcanonical simulations limiting the energy E to some small range (E_i, E_{i+1}). For each simulation, one would count the number of times a given energy would appear, leading to a direct estimate of the ratios $\Omega(E)/\Omega(E')$ for $E, E' \in (E_i, E_{i+1})$. Repeating the simulations for all possible ranges of energies, Bhanot and collaborators were able to compute numerically approximations to $\Omega(E)$, and hence to $S(E)$, for all values of E in the 3-D Ising and a variety of other models.

4) The name multicanonical appears because $e^{-S_{\text{app}}(X)}$ can be written trivially as $e^{-\beta(X)\mathcal{H}(X)}$ with $\beta(X) = S_{\text{app}}(X)/\mathcal{H}(X)$, an effective inverse temperature depending on the energy of the configuration.

Another approach is that of Berg and collaborators [85–87], which uses a recursive procedure. The idea is to use a first choice $\omega^{(0)}(E) = 0$ (this is equivalent to a canonical algorithm at $T = \infty$). One then obtains a first estimate for the entropy $S^{(0)}(E)$ using (F.5). However, this formula cannot be used for those values of E that did not appear during the simulation, as the histogram count $H_\omega(E)$ is zero for them and one cannot compute log(0). In the next step, one chooses a function $\omega^{(1)} = \omega^{(0)}(E)$ in the region visited in the previous simulation and zero elsewhere. This leads to an estimate for the entropy $S^{(1)}(E)$, valid in a larger range of energies, and so on. This recurrence procedure stops when the energy range in which the rth recurrence entropy $S^{(r)}(E)$ is valid covers the desired range $(E_{\min}, E_{\max})$. Some special care is needed to ensure convergence of the recurrence process, and Berg introduces an explicit recurrence scheme that allows systematic corrections of $\omega^{(i)}(E)$ at all visited energies.

This procedure has some limitations. To begin with, the entropy can only be estimated inside the energy range visited in the last simulation, and a bad choice for the entropy outside this region can severely limit the exploration of lower energies. Secondly, rarely visited energies introduce a large error in the estimated entropy (hence the need for recurrence introduced by Berg in order to minimize this error). Furthermore, the fact that one counts visits in each energy implies that for continuous systems the energy spectrum must be discretized.

The celebrated method of Wang and Landau [88–90] uses a different approach in which the function $\omega(E)$ is constantly changing until it converges to the desired entropy function $S(E)$. Without going into fine details, the idea is to start with the choice $\omega(E) = 0$, $\forall E$, and then sample the distribution (F.4) using, for example, the Metropolis algorithm. The new ingredient is that, every time a configuration X of energy E is accepted, then this particular value of E modifies the function $\omega(E) \to \omega(E) + \delta$ with $\delta = 1$. After some equilibration steps of this sort, the process continues but now the quantity added to $\omega(E)$ for each accepted value of the energy is $\delta = 1/2$. At the next set or runs, it sets $\delta = 1/4$, and so on. The process continues until a very small value for δ is reached, say $\delta = 10^{-8}$. At this time, the function $\omega(E)$ has converged to the entropy $S(E)$ (except by an additive constant) and the histogram $H_\omega(E)$ is flat. It is intuitive that the histogram should be flat. When we add a number δ, we are reducing by a factor $e^{-\delta}$ the probability of that configuration, so that more probable configurations are given a smaller chance to appear. There are some technical problems to make this an effective procedure [91], but there are now a very large number of papers that have used this technique to a large variety of problems, and it has become one of the standard procedures in the field of multicanonical sampling.

We would like to end this section with another approach that has proven successful in some cases and can be easily implemented. We follow here closely the presentation of [92]. This approach assumes that thermodynamic functions such as the entropy can be treated as continuous functions of energy, both for discrete and continuous spectra, for not too small systems. More specifically, one

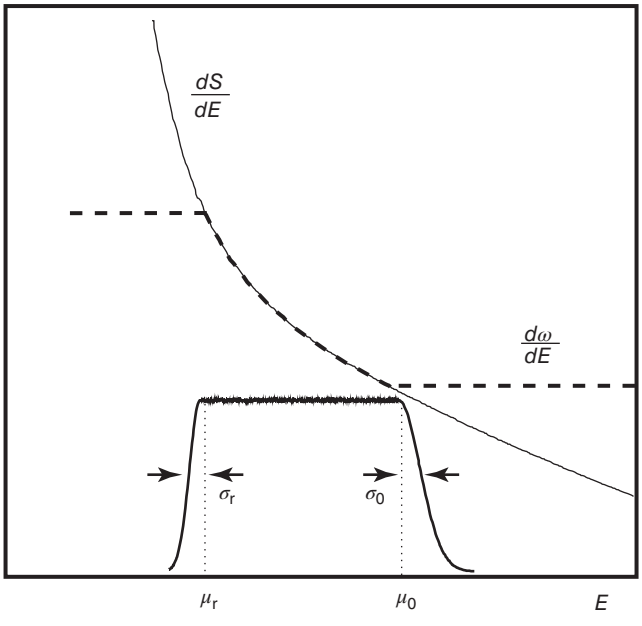


Figure F.1 In the current proposal, the weight function $\omega(E)$ approximates the entropy between two energies μ_r and μ_0 and is linear in energy outside this region, $d\omega/dE = \beta_r$ for $E < \mu_r$ and $d\omega/dE = \beta_0$ for $E > \mu_0$. (*Source:* Reprinted from [92]).

makes use of the Gaussian approximation

$$S(E) \approx S(\mu_\beta) + \beta(E - \mu_\beta) - \frac{1}{2\sigma_\beta^2}(E - \mu_\beta)^2 \tag{F.11}$$

to propose the sequence for the weight functions $\omega(E)$. The procedure is such that, at each recurrence step, the distribution $H_\omega(E)$ is flattened in a region centered around the mean, and the width is proportional to the standard deviation of the energy distribution at the previous step.

The scheme works as follows: after running an initial simulation at inverse temperature β_0, with $\omega_0 = \beta_0 E$, we measure the mean value μ_0 and the variance σ_0^2 of the resulting energy distribution. We then modify the weight function to $\omega_1(E)$ defined by

$$\omega_1(E) = \begin{cases} \beta_0 E, & E > \mu_0, \\ b_0 + \beta_0(E - \mu_0) - \dfrac{(E - \mu_0)^2}{2\sigma_0^2}, & \mu_1 < E < \mu_0, \\ b_1 + \beta_1(E - \mu_1), & E < \mu_1, \end{cases} \tag{F.12}$$

with $\mu_1 = \mu_0 - \alpha\sigma_0$, and α a number of order 1. In order to ensure continuity of the function and its derivative, we determine $b_0 = \beta_0\mu_0$, $b_1 = \beta_0\mu_1 - \frac{\alpha^2}{2}$, and $\beta_1 = \beta_0 + \alpha/\sigma_0$. In a new simulation with transition rates $W_{ij} = \min(1, e^{-\Delta\omega_1})$, we obtain $H_\omega(E) \simeq$ constant, for energies such that $\mu_1 < E < \mu_0$, while for $E < \mu_1$, $H(E)$

is a ''half Gaussian'' with maximum at μ_1 and half-width σ_1. We now compute σ_1^2 as the average of $\left(E-\mu_1\right)^2$ for $E<\mu_1$. We are therefore able to add another branch to $\omega(E)$

$$\omega_2(E)=\begin{cases}\beta_0 E & E>\mu_0\\ b_0+\beta_0(E-\mu_0)-\dfrac{(E-\mu_0)^2}{2\sigma_0^2}, & \mu_1<E<\mu_0,\\ b_1+\beta_1(E-\mu_1)-\dfrac{(E-\mu_1)^2}{2\sigma_1^2}, & \mu_2<E<\mu_1,\\ b_2+\beta_2(E-\mu_2), & E<\mu_2\end{cases} \tag{F.13}$$

with $\mu_2=\mu_1-\alpha\sigma_1$, $b_2=\beta_1\mu_2-\frac{\alpha^2}{2}$, and $\beta_2=\beta_1+\alpha/\sigma_1$. This process can be repeated with similar recurrence relations for the coefficients until we reach the lowest temperature we wish to study. On the iteration of order r, the histogram is nearly flat between μ_0 and μ_r and half Gaussian below μ_r (see Figure F.1). Although the entropy could also be extrapolated to higher values of the energy, using $\omega(E)=\beta_0 E$ for $E>\mu_0$, we effectively restrict the simulation to energies below μ_0, apart from a Gaussian tail above this energy.

Appendix G
Discrete Fourier Transform

Fourier series and integrals are very powerful analytical tools that are used in many problems. From the numerical point of view, when using for instance finite differences to solve numerically a partial differential equation or in many other cases of interest, we only know the relevant function at selected equidistant points in space. The approximation of the integral by a sum over the selected points leads naturally to the concept of discrete Fourier transform. In this appendix, we explain the relation between a *Fourier series* and a *discrete Fourier transform* and the errors associated to the approximation. We will also explain the use of fast Fourier routines to compute efficiently the discrete Fourier transform.

G.1
Relation Between the Fourier Series and the Discrete Fourier Transform

We consider (real or complex) functions $f(\vec{x})$ defined in a finite d-dimensional volume[1] $\vec{x} = (x_1, \ldots, x_d) \in [0, L]^d$. We extend the definition to all points $\vec{x} \in \mathbb{R}^d$ using the periodicity conditions

$$f(\vec{x} + L\hat{\mathrm{e}}_i) = f(\vec{x}), \qquad i = 1, \ldots, d, \tag{G.1}$$

where $\hat{\mathrm{e}}_i$ is the unitary vector in space-direction i: $\hat{\mathrm{e}}_1 = (1, 0, 0, \ldots)$, $\hat{\mathrm{e}}_2 = (0, 1, 0, \ldots)$, etc. The Fourier series for these periodic functions is defined as

$$\tilde{f}(\vec{q}) = \mathcal{F}_P[f(\vec{x})] \equiv \frac{1}{L^d} \int_{[0,L]^d} d\vec{x}\, f(\vec{x}) \mathrm{e}^{\mathrm{i}\vec{q}\cdot\vec{x}}, \quad \mathrm{i} = \sqrt{-1}. \tag{G.2}$$

A definition valid $\forall \vec{q} \in \mathbb{R}^d$. We introduce the following notation:

$$\tilde{f}_{\vec{k}} = \tilde{f}(\vec{q}_{\vec{k}}), \qquad \vec{q}_{\vec{k}} = \frac{2\pi}{L}\vec{k}, \tag{G.3}$$

1) We assume that each of the d coordinates expands the same limits $[0, L]$. This is always possible by a linear transformation of the original coordinates.

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.

and $\vec{k} = (k_1, \ldots, k_d)$. It is well known that (given some general conditions) the following (inverse) Fourier series is an exact representation of $f(\vec{x})$[2]:

$$f(\vec{x}) = \mathcal{F}_P^{-1}[\tilde{f}(\vec{q})] \equiv \sum_{\vec{k}=-\infty}^{\infty} \tilde{f}_{\vec{k}} e^{-i\frac{2\pi}{L}\vec{k}\cdot\vec{x}}, \tag{G.4}$$

valid for all points $\vec{x} \in \mathbb{R}^d$. Therefore, the knowledge of the periodic function $f(\vec{x})$ or the (infinite) set of coefficients $\tilde{f}_{\vec{k}}$ can be considered as equivalent as one can be obtained from the other. It is usually very difficult to compute the $\tilde{f}_{\vec{k}}$'s because their definition involves a d-dimensional integration. Therefore, the *first approximation* useful for numerical calculations is to compute them using only the values of the function $f(\vec{x})$ at selected lattice points $\vec{x}_{\vec{n}} = \vec{n}\Delta x$, for $n = (n_1, \ldots, n_d)$ and $n_i = 0, \ldots, N-1$.[3] The physical length is $L = N\Delta x$. We compute the coefficients of the Fourier series using only $f_{\vec{n}} \equiv f(\vec{x}_{\vec{n}})$ and approximate (G.2) by

$$\tilde{f}_{\vec{k}} \approx \frac{1}{N^d}\hat{f}_{\vec{k}}, \tag{G.5}$$

where we have defined define the discrete Fourier transform as

$$\hat{f}_{\vec{k}} = \mathcal{F}_D[f_{\vec{n}}] \equiv \sum_{\vec{n}=0}^{N-1} f_{\vec{n}} e^{i\frac{2\pi}{N}\vec{k}\cdot\vec{n}}. \tag{G.6}$$

If $f(\vec{x})$ is a real function, we note the property:

$$\hat{f}_{\vec{k}}^* = \hat{f}_{-\vec{k}}. \tag{G.7}$$

The *second approximation* is that the infinite sum in (G.4) is replaced by a finite one using the discrete coefficients:

$$f(\vec{x}) \approx \overline{f}(\vec{x}) \equiv \frac{1}{N^d} \sum_{\vec{k}=\Lambda_1}^{\Lambda_2} \hat{f}_{\vec{k}} e^{-i\frac{2\pi}{L}\vec{k}\cdot\vec{x}}. \tag{G.8}$$

The nice thing of these two approximations (G.5) and (G.8) is that we can demand that they cancel each other such the previous formula becomes then exact for the lattice points $\vec{x}_{\vec{n}}$, that is, $f(\vec{x}_{\vec{n}}) = \overline{f}(\vec{x}_{\vec{n}})$. This is achieved using any set of values Λ_1, Λ_2 such that $\Lambda_2 - \Lambda_1 = N - 1$. Therefore, we define the discrete inverse Fourier transform as

$$\mathcal{F}_D^{-1}[\hat{f}_{\vec{k}}] \equiv \frac{1}{N^d} \sum_{\vec{k}=\Lambda_1}^{\Lambda_1+N-1} \hat{f}_{\vec{k}} e^{-i\frac{2\pi}{N}\vec{k}\cdot\vec{n}}. \tag{G.9}$$

2) This and other sums have to be understood as each of the components of $\vec{k} = (k_1, \ldots, k_d)$ or $\vec{n} = (n_1, \ldots, n_d)$ varying between the lower and upper limit:

$$\sum_{\vec{k}=\Lambda_1}^{\Lambda_2} \equiv \sum_{k_1=\Lambda_1}^{\Lambda_2} \cdots \sum_{k_d=\Lambda_1}^{\Lambda_2}, \qquad \text{or} \qquad \sum_{\vec{n}=N_1}^{N_2} \equiv \sum_{n_1=N_1}^{N_2} \cdots \sum_{n_d=N_1}^{N_2}.$$

3) It is possible to use a different number of points $N_1, N_2, \ldots, N_d$ for every spatial dimension. We will only use this possibility only when explaining the general calls to the numerical routines, but prefer to consider a common value N and Δx for the sake of simplicity in the notation.

Indeed, using the property

$$\frac{1}{N^d}\sum_{\vec{n}} e^{i\frac{2\pi}{N}\vec{k}\cdot\vec{n}} = \delta_{\vec{k},0}, \tag{G.10}$$

it is possible to prove that whatever the value of Λ_1 we have

$$\mathcal{F}_D^{-1}[\hat{f}_{\vec{k}}] = f(\vec{x}_{\vec{n}}) = f_{\vec{n}}. \tag{G.11}$$

From this point of view we can think of the formulas (G.6) and (G.9) as a one-to-one correspondence between the sets of coefficients $\{f_{\vec{n}}\}$ and $\{\hat{f}_{\vec{k}}\}$ with $\vec{n} = (n_1, \ldots, n_d)$, $n_i = 0, \ldots, N-1$ and $\vec{k} = (k_1, \ldots, k_d)$, $k_i = 0, \ldots, N-1$. As far as this correspondence is concerned, the exact value of Λ_1 is irrelevant.

We now pose the question of which are the most convenient values of Λ_1 and Λ_2. Since we just said that any values such that $\Lambda_2 - \Lambda_1 = N-1$ would imply the important property (G.11) that allows one to recover the original values of $f(\vec{x})$ at the points $\vec{x} = \vec{x}_n$, what do we mean exactly by choosing the right values for Λ_1 and Λ_2? The answer is that we mean that the function $\overline{f}(\vec{x})$ given by (G.8) is a good approximation to $f(\vec{x})$ **also at other values of** $\vec{x}$.

To choose Λ_1 and Λ_2 in order to obtain the best approximation of $\overline{f}(\vec{x})$ to $f(\vec{x})$ we have to understand a little bit more deeply the relation between the coefficients $\tilde{f}_{\vec{k}}$ of the Fourier series and the coefficients $\hat{f}_{\vec{k}}$ of the discrete Fourier transform. The way we have defined them implies that the $\hat{f}_{\vec{k}}$ are an approximation, (G.5), to the $\tilde{f}_{\vec{k}}$. How good an approximation? It turns out that the approximation is very bad for large values of $|\vec{k}|$. And the reason is due to a property of the $\hat{f}_{\vec{k}}$ called "aliasing" which is not present in the coefficients $\tilde{f}_{\vec{k}}$. The property, which is easily derived from their definition (G.6), states that the coefficients of the discrete Fourier transform are periodic with a period of N in the indexes:

$$\hat{f}_{\vec{k}+N\hat{e}_i} = \hat{f}_{\vec{k}}, \qquad i = 1, \ldots, d. \tag{G.12}$$

To simplify notation, from now on we switch to $d = 1$. In this case, (G.12) states, for example, that $\hat{f}_N = \hat{f}_0$, whereas the coefficients of the Fourier series $\tilde{f}_k$ for a well-behaved function decrease with k for large k. In practice, it means that the approximation (G.5) of $\hat{f}_k$ to $\tilde{f}_k$ begins to worsen for $|k| > N/2$ (see Figure G.1 later). For $|k| > N/2$ the $\hat{f}_k$ are given the values of their "alias" modes. For instance, when $N = 8$, $\hat{f}_{18} = \hat{f}_{10} = \hat{f}_2 = \hat{f}_{-6} \ldots$. But we can only affirm that $\tilde{f}_2 \approx \hat{f}_2/N$, we cannot say that $\tilde{f}_{10} \approx \hat{f}_{10}/N$ or $\tilde{f}_{18} \approx \hat{f}_{18}/N$. This result and the fact that the Fourier series (G.4) involves an infinite sum of positive and negative terms, leads us to keep in the truncated Fourier series (G.8) a symmetric interval for (Λ_1, Λ_2). If $N = 2M+1$ is an odd number, the obvious choice is $(\Lambda_1, \Lambda_2) = (-M, M)$ and the series is

$$f(x) \approx \frac{1}{N^d}\sum_{k=-M}^{M} \hat{f}_k e^{-i\frac{2\pi}{L}kx}. \tag{G.13}$$

If $f(x)$ is a real function, this choice (and any other satisfying $\Lambda_1 = -\Lambda_2$) has the additional advantage that the imaginary part of (G.8) is exactly zero for all values

of x, so we do not introduce a fake complex imaginary value in the approximation $\overline{f}(x)$. Let us prove that (G.8) is real for $\Lambda_1 = -\Lambda_2$:

$$\left(\sum_{k=-\Lambda_2}^{\Lambda_2} \hat{f}_k e^{-i\frac{2\pi}{L}kx}\right)^* = \sum_{k=-\Lambda_2}^{\Lambda_2} \hat{f}_k^* e^{i\frac{2\pi}{L}kx} = \sum_{k=-\Lambda_2}^{\Lambda_2} \hat{f}_{-k} e^{i\frac{2\pi}{L}kx} = \sum_{k=-\Lambda_2}^{\Lambda_2} \hat{f}_k e^{-i\frac{2\pi}{L}kx}. \tag{G.14}$$

If $N = 2M$ is an even number (the commonest case in practice due to the fact that fast Fourier routines are more effective and the only one considered henceforth), one can not fulfill the requirements $\Lambda_1 = -\Lambda_2$ and $\Lambda_2 - \Lambda_1 = N - 1$. In this case there are two equally reasonable choices $(\Lambda_1, \Lambda_2) = (-M + 1, M)$ or $(\Lambda_1, \Lambda_2) = (-M, M - 1)$. In the first option, we write the Fourier series as

$$f(x) \approx \overline{f}_1(x) \equiv \frac{1}{N^d} \sum_{k=-M+1}^{M} \hat{f}_k e^{-i\frac{2\pi}{L}kx}. \tag{G.15}$$

However, for a real function $f(x)$, this expression has the disadvantage that, in general, it yields a nonzero imaginary part for arbitrary values of x. The best alternative in the case of N even is to use a truly symmetric choice, namely:

$$f(x) \approx \overline{f}_2(x) \equiv \frac{1}{N^d} \left[\frac{1}{2}\hat{f}_{-M} e^{i\frac{2\pi}{L}Mx} + \sum_{k=-M+1}^{M-1} \hat{f}_k e^{-i\frac{2\pi}{L}kx} + \frac{1}{2}\hat{f}_M e^{-i\frac{2\pi}{L}Mx}\right], \tag{G.16}$$

which will be written as

$$f(x) \approx \overline{f}_2(x) \equiv \frac{1}{N^d} {\sum_{k=-M}^{M}}' \hat{f}_k e^{-i\frac{2\pi}{L}kx}, \tag{G.17}$$

where the prime in the sum indicates that the first and last terms are halved. This symmetric option is a little bit more messy to implement in practice although it has the advantage that the imaginary part is exactly zero. The only difference between the expressions (G.15) and (G.17) is in the imaginary part. In fact if we subtract (G.15) from (G.17), we obtain:

$$\frac{i\hat{f}_M}{N^d} \sin\left(\frac{2\pi}{L}Mx\right). \tag{G.18}$$

In the case $N = 2M$ we can prove using (G.7) and (G.12) that $\hat{f}_{-M} = \hat{f}_M$ is real and, therefore, this last expression is purely imaginary (although at the lattice points $x_n = n\Delta x$ this imaginary part vanishes). Therefore, an alternative expression to (G.17) valid for real functions $f(x)$ and somewhat more useful for the numerical programming is to take the real part of (G.15):

$$\overline{f}_2(x) = \mathcal{R}\left[\frac{1}{N^d} \sum_{k=-M+1}^{M} \hat{f}_k e^{-i\frac{2\pi}{L}kx}\right]. \tag{G.19}$$

We use as an example the function $f(x) = 3/(5 - 4\cos x)$. This function is periodic of period $L = 2\pi$. The coefficients of the Fourier series can be calculated analytically with the result $\tilde{f}_k = 2^{-|k|}$. The coefficients of the discrete Fourier

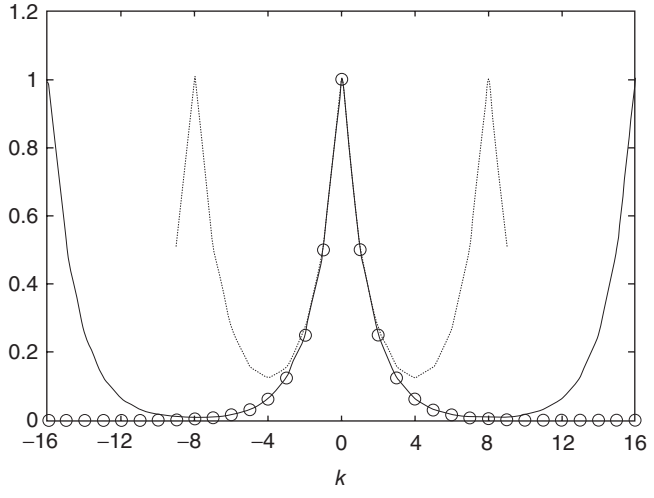


Figure G.1 Comparison between the coefficients of the Fourier series, $\tilde{f}_k$ (symbols), and those of the discrete Fourier series, $\hat{f}_k/N$ (lines), for the function $f(x) = 3/(5 - 4\cos x)$. The dotted line corresponds to $N = 8$ and the solid line to $N = 16$. Notice that, due to aliasing, the discrete coefficients are a good approximation to the coefficients of the (infinite) Fourier series only for $|k| \lesssim N/2$.

transform, $\hat{f}_k$, depend on the number of points N used in the discretization of $(0, 2\pi)$. Figure (G.1) shows the difference between the coefficients $\hat{f}_k$ and $\tilde{f}_k$ for different values of N. It is possible to observe in this figure the effect of aliasing. Only for $|k| < N/2$ are the coefficients $\hat{f}_k/N$ a reasonable approximation to $\tilde{f}_k$. Note also that the coefficients $\hat{f}_k$ satisfy the periodicity condition (G.12) while $\tilde{f}_k$ decay for $|k|$ large.

In Figure G.2, we compare the function and the two approximations, $\overline{f}_1(x)$ and $\overline{f}_2(x)$ in the case $N = 8$. Note that, although both are reasonable approximations and both give the exact values for $f(x_n)$, $\overline{f}_1(x)$ has a nonzero imaginary part, as shown in Figure G.3, while $\overline{f}_1(x)$ is always real. Just to warn the reader not to use the approximation (G.8) with $\Lambda_1 = 0$ (as one would do when using carelessly the coefficients in the same order that is given by a fast Fourier transform routine, see next section), we plot

$$\overline{f}_3(x) = \frac{1}{N} \sum_{k=0}^{N-1} \hat{f}_k e^{-i\frac{2\pi}{L}kx}, \tag{G.20}$$

to show that, due to aliasing, it introduces a big error at points other than x_n. For the sake of comparison we have also included the approximation that uses (G.17) but using instead of $\hat{f}_k$ the exact Fourier coefficients, namely:

$$\overline{f}_4(x) = \sum_{k=-M}^{M}{}' \tilde{f}_k e^{-i\frac{2\pi}{L}kx}. \tag{G.21}$$

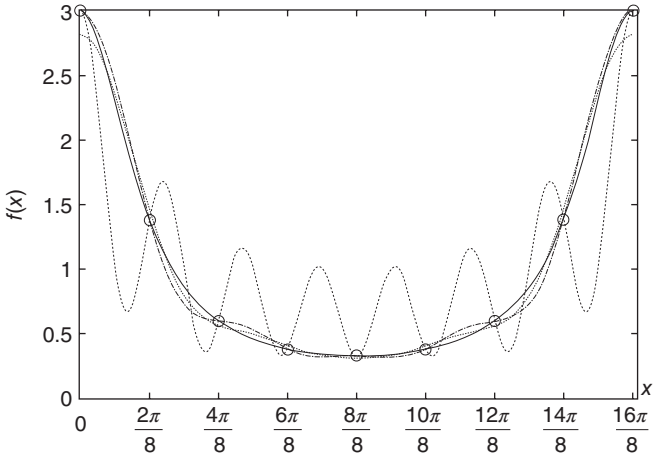


Figure G.2 Comparison between the function $f(x) = 3/(5 - 4\cos x)$, solid line and points, and its different Fourier approximations in the case $N = 8$: dot–dashed (G.17), dashed (G.20), and dotted (G.21).

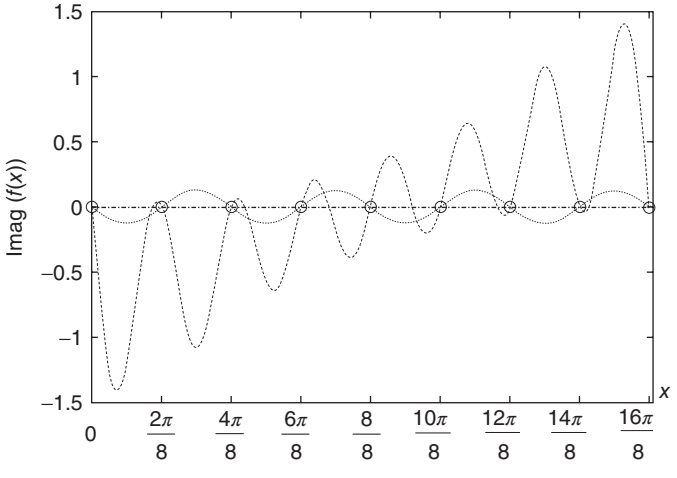


Figure G.3 Imaginary part of the Fourier approximations to the $f(x) = 3/(5 - 4\cos x)$ in the case $N = 8$: dotted (G.15), dashed (G.20).

The message is that, in the case of even $N = 2M$, one has to use either approximation (G.15) or (G.17). The second one is better because the imaginary part of this approximation for real functions is always equal to zero. Alternatively, one could use the approximation (G.21) which uses the exact coefficients, but it is very rare in practice to know the exact coefficients. The calculation of the coefficients $\hat{f}_k$ can be done using very efficient algorithms that we will explain in Section G.3.

G.2 Evaluation of Spatial Derivatives

In pseudospectral methods for partial differential equations one has a field $f(x,t)$, whose spatial derivatives need to be evaluated in Fourier space. For one spatial dimension, considering now f to be a function of x and t, from (G.17) we have

$$\frac{\partial f(x,t)}{\partial x} \approx \frac{1}{N} \sum_{k=-M}^{M}{}' \left(-i\frac{2\pi}{L}k\right) \hat{f}_k(t) \mathrm{e}^{-i\frac{2\pi}{L}kx}. \tag{G.22}$$

Since $\hat{f}_M = \hat{f}_{-M}$ the first and last terms (the halved ones) cancel and we obtain:

$$\frac{\partial f(x,t)}{\partial x} \approx \frac{1}{N} \sum_{k=-M+1}^{M-1} \left(-i\frac{2\pi}{L}k\right) \hat{f}_k(t) \mathrm{e}^{-i\frac{2\pi}{L}kx}, \tag{G.23}$$

which we can write as

$$\frac{\partial f(x,t)}{\partial x} \approx \frac{1}{N} \sum_{k=-M+1}^{M-1} -iq_k \hat{f}_k(t) \mathrm{e}^{-i\frac{2\pi}{L}kx}, \tag{G.24}$$

with

$$\begin{aligned} q_k &= \frac{2\pi}{L}k, \qquad k = -\frac{N}{2}+1, \ldots, \frac{N}{2}-1 \\ q_{\frac{N}{2}} &= 0. \end{aligned} \tag{G.25}$$

Similarly, higher-order spatial derivatives can be evaluated in Fourier space as

$$\frac{\partial^n f(x,t)}{\partial x^n} \approx \frac{1}{N} \sum_{k=-M+1}^{M-1} (-iq_k)^n \hat{f}_k(t) \mathrm{e}^{-i\frac{2\pi}{L}kx}. \tag{G.26}$$

For a field in d-spatial dimensions, one can proceed in a similar way

$$\frac{\partial^n f(x,t)}{\partial x_j^n} \approx \frac{1}{N^d} \sum_{\vec{k}=-M+1}^{M-1} (-iq_{k_j})^n \hat{f}_k(t) \mathrm{e}^{-i\frac{2\pi}{L}\vec{k}\vec{x}}, \tag{G.27}$$

with

$$\begin{aligned} \vec{q}_{\vec{k}} &= \frac{2\pi}{L}\vec{k}, \qquad k_j = -\frac{N}{2}+1, \ldots, \frac{N}{2}-1, \qquad j = 1, \ldots, d \\ \vec{q}_{N/2} &= 0. \end{aligned} \tag{G.28}$$

G.3 The Fast Fourier Transform

There is one nice thing about the discrete Fourier transforms $\mathcal{F}_D$ and $\mathcal{F}_D^{-1}$. They satisfy some recurrence relations that allow them to be computed very efficiently with convenient numerical algorithms. These algorithms are known under the general name of *fast Fourier transforms*, but one has to realize that they are nothing new, just a way of organizing the sums in order to compute the coefficients

$\{\hat{f}_{\vec{k}}\}$ starting from $\{f_{\vec{n}}\}$ (direct fast Fourier transform) and vice-versa (inverse fast Fourier transform). The algorithms are particularly efficient when N is a power of 2 and some implementations of the fast Fourier transforms demand this condition explicitly, while others are more flexible and work reasonably well with any value of N, but in general the efficiency improves if N has a large number of divisors (and it is worst when N is a prime number). In general, the time needed by a fast Fourier transform routine scales as $N^d \log N$ A very important thing to care about is that there is not a general agreement about the *exact* definition of the discrete Fourier transform. The different definitions introduce different factors of N before the sums. Namely:

$$\hat{f}_{\vec{k}} = \mathcal{F}_D[f_{\vec{n}}] \equiv \frac{1}{N^a} \sum_{\vec{n}=0}^{N-1} f_{\vec{n}} e^{i\frac{2\pi}{N}\vec{k}\cdot\vec{n}}, \tag{G.29}$$

$$f_{\vec{n}} = \mathcal{F}_D^{-1}[\hat{f}_{\vec{k}}] \equiv \frac{1}{N^b} \sum_{\vec{k}=0}^{N-1} \hat{f}_{\vec{k}} e^{-i\frac{2\pi}{N}\vec{k}\cdot\vec{n}}. \tag{G.30}$$

As far as $a + b = d$ it is still true that they are the inverse of each other. We have adopted here the values $a = 0$, $b = d$, but other choices are possible and, more importantly, they are commonly used. For instance, most fast Fourier routines take $a = b = d/2$,[4] but others take $a = d$, $b = 0$. To add confusion, some routines take $a = b = 0$ (so it is no longer true that they are inverse of each other) but they expect YOU to correct the factors of N afterward. There is no other solution to this lack of uniformity than to read carefully the definition adopted by the particular algorithm we are using.

In general, the routines take as input a d-dimensional complex vector[5]

```
double complex f(0:N1-1,...,0:Nd-1)
```

holding the numbers $f_{\vec{n}}$[6] and return a d-dimensional complex vector of the same dimension with the coefficients $\hat{f}_{\vec{k}}$. Most times, to save memory, the coefficients $\hat{f}_{\vec{k}}$ overwrite the $f_{\vec{n}}$. A typical call is then:

```
call fft(f,N,+1)
```

where `N` is a vector with d components such that `N(1)`$=N_1, \ldots,$ `N(d)`$=N_d$. The `+1` indicates that this is a direct transform: the input vector `f` contains $f_{\vec{n}}$ and the output $\hat{f}_{\vec{k}}$ is overwritten in `f`. The inverse call

```
call fft(f,N,-1)
```

is such that on input vector `f` contains $\hat{f}_{\vec{k}}$ and on output it contains $f_{\vec{n}}$. Two successive calls:

4) This choice has the additional that the Jacobian of the change of variables $\{f_{\vec{n}}\} \to \{\hat{f}_{\vec{k}}\}$ is equal to 1.
5) It is common the use of the double precision. Again, read the manual of the fast Fourier routines!
6) We accept here the possibility that each spatial dimension is discretized by a different number of points.

```
call fft(f,N,+1)
```

```
call fft(f,N,-1)
```

should leave `f` unchanged (or at least with all its entries multiplied with the same power of N, should the routine be written this way).

Usually there is a specific call for $d = 1$, something like

```
call fft1d(f,N,+1)
```

and the inverse call

```
call fft1d(f,N,-1)
```

where now vector `f` has only one index, `double complex f(0:N-1)`, and `N` is an integer number. Remember the aliasing problem: (let us take $N = 8$) if the elements `f(0),f(1),f(2),f(3),f(4),f(5),f(6),f(7)` contain, on input to `call fft1d(f,N,+1)`, the values of $f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7$, then the output should not be considered as containing $\hat{f}_0, \hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4, \hat{f}_5, \hat{f}_6, \hat{f}_7$, but $\hat{f}_0, \hat{f}_1, \hat{f}_2, \hat{f}_3, \hat{f}_4, \hat{f}_{-3}, \hat{f}_{-2}, \hat{f}_{-1}$.

An important case is when the input f_n is a vector or real numbers. Then the output $\hat{f}_k$ satisfies (G.7) which implies that if we write $\hat{f}_k = r_k + is_k$, not all numbers r_k, s_k are independent of each other.[7] Take, for instance, $N = 8$. As $\hat{f}_7 = \hat{f}_1^*$, $\hat{f}_6 = \hat{f}_2^*$, $\hat{f}_5 = \hat{f}_3^*$, as well as $\hat{f}_0 = \hat{f}_0^*$, $\hat{f}_4 = \hat{f}_4^*$, the only independent variables are $r_0, r_1, s_1, r_2, s_2, r_3, s_3, r_4$. To save computer memory by not storing the coefficients which are not needed, most (read the manual!) fast Fourier transforms of a real vector would take as input

$$(f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7)$$

as the components of vector `double precision f(0:7)` and overwrite with

$$(r_0, r_4, r_1, s_1, r_2, s_2, r_3, s_3),$$

in that precise order (and a similar generalization for arbitrary n even). This could be implemented in a call such as

```
double precision f(0:N-1)
```

```
call realfft1d(f,N,+1)
```

There are similar simplifications for spatial dimension $d > 1$, but they are a little bit more cumbersome and will not be explained here.

Further Reading

The review [59] discusses aliasing and some questions about the use of Fourier techniques in solving numerically differential equations.

7) This was also explained in Appendix B.

Index

Stochastic Numerical Methods: An Introduction for Students and Scientists, First Edition.
Raúl Toral and Pere Colet.