Numerical Methods in Finance and Economics

A MATLAB-Based Introduction

Second Edition

Paolo Brandimarte
Politecnico di Torino
Torino, Italy

WILEY-INTERSCIENCE
A JOHN WILEY & SONS, INC., PUBLICATION

This Page Intentionally Left Blank

Numerical Methods in Finance and Economics

Numerical Methods in Finance and Economics

A MATLAB-Based Introduction

Second Edition

Paolo Brandimarte

Politecnico di Torino
Torino, Italy

WILEY-INTERSCIENCE

A JOHN WILEY & SONS, INC., PUBLICATION

Library of Congress Cataloging-in-Publication Data:

Brandimarte, Paolo.
Numerical methods in finance and economics : a MATLAB-based introduction / Paolo Brandimarte.—2nd ed.
p. cm.
Rev. ed. of: Numerical methods in finance. 2002.
Includes bibliographical references and index.
ISBN-13: 978-0-471-74503-7 (cloth)
ISBN-10: 0-471-74503-0 (cloth)
1. Finance—Statistical methods. 2. Economics—Statistical methods. I. Brandimarte, Paolo. Numerical methods in finance. II. Title.
HG176.5.B73 2006
332.01'51—dc22 2006045787

Printed in the United States of America.

10 9 8 7 6 5 4 3 2 1

This book is dedicated to Commander Straker, Lieutenant Ellis, and all SHADO operatives. Thirty-five years ago they introduced me to the art of using both computers and *gut feelings to make decisions.*

This Page Intentionally Left Blank

Contents

This Page Intentionally Left Blank

Preface to the Second Edition

After the publication of the first edition of the book, about five years ago, I have received a fair number of messages from readers, both students and practitioners, around the world. The recurring keyword, and the most important thing to me, was *useful.* The book had, and has, no ambition of being a very advanced research book. The basic motivation behind this second edition is the same behind the first one: providing the newcomer with an easy, but solid, entry point to computational finance, without too much sophisticated mathematics and avoiding the burden of difficult C++ code, also covering relatively non-standard optimization topics such as stochastic and integer programming. See also the excerpt from the preface to the first edition. However, there are a few new things here:

- a slightly revised title;
- completely revised organization of chapters;
- significantly increased number of pages.

The title mentions both Finance *and* Economics, rather than just Finance. To avoid any misunderstanding, it should be made quite clear that this is essentially a book for students and practitioners working in Finance. Nevertheless, it *can* be useful to Ph.D. students in Economics as well, as a complement to more specific and advanced textbooks. In the last four years, I have been giving a course on numerical methods within a Ph.D. program in Economics, and I typically use other available excellent textbooks covering advanced algorithms[1] or offering well-thought MATLAB toolboxes[2] which can be used to solve a wide array of problems in Economics. From the point of view of my students in such a course, the present book has many deficiencies: For instance, it does not cover ordinary differential equations and it does not deal with computing equilibria or rational expectations models; furthermore, practically all of the examples deal with option pricing or portfolio management. Nevertheless, given my experience, I believe that they can benefit from a more detailed and elementary treatment of the basics, supported by simple examples. Moreover, I believe that students in Economics should also get

[1]K.L. Judd, *Numerical Methods in Economics*, MIT Press, 1998.
[2]M.J. Miranda and P.L. Fackler, *Applied Computational Economics and Finance*, MIT Press, 2002.

at least acquainted with topics from Operations Research, such as stochastic programming and integer programming. Hence, the "*and Economics*" part of the title suggests potential use of the book as a complement, and by no means as a substitute.

The book has been reorganized in order to ease its use within standard courses on numerical methods for financial engineering. In the first edition, optimization applications were dealt with extensively, in chapters preceding those related to option pricing. This was a result of my personal background, which is mainly Computer Science and Operations Research, but it did not fit very well with the common use of a book on computational finance. In the present edition, advanced optimization applications are left to the last chapters, so they do not get into the way of most financial engineering students. The book consists of twelve chapters and three appendices.

- Chapter 1 provides the reader with motivations for the use of numerical methods, and for the use of MATLAB as well.

- Chapter 2 is an overview of financial theory. It is aimed at students in Engineering, Mathematics, or Operations Research, who may be interested in the book, but have little or no financial background.

- Chapter 3 is devoted to the basics of classical numerical methods. In some sense, this is complementary to chapter 2 and it is aimed at people with a background in Economics, who typically are not exposed to numerical analysis. To keep the book to a reasonable size, a few classical topics were omitted because of their limited role in the following chapters. In particular, I do not cover computation of eigenvalues and eigenvectors and ordinary differential equations.

- Chapter 4 is devoted to numerical integration, both by quadrature formulas and Monte Carlo methods. In the first edition, quadrature formulas were dealt with in the chapter on numerical analysis, and Monte Carlo was the subject of a separate chapter. I preferred giving a unified treatment of these two approaches, as this helps understanding their respective strengths and weaknesses, both for option pricing and scenario generation in stochastic optimization. Regarding Monte Carlo as a tool for integration rather than simulation is also helpful to properly frame the application of low-discrepancy sequences (which is also known under the more appealing name of quasi-Monte Carlo simulation). There is some new material on Gaussian quadrature, an extensive treatment of variance reduction methods, and some application to vanilla options to illustrate simple but concrete applications immediately, leaving more complex cases to chapter 8.

- Chapter 5 deals with basic finite difference schemes for partial differential equations. The main theme is solving the heat equation, which is the prototype example of the class of parabolic equations, to which Black–Scholes equation belongs. In this simplified framework we may understand the difference between explicit and implicit methods, as well as the issues related to convergence and numerical stability. With respect to the first edition, I have added an outline of the Alternating Direction Implicit method to solve the two-dimensional heat equation, which is useful background for pricing multidimensional options.

- Chapter 6 deals with finite-dimensional (static) optimization. This chapter can be safely skipped by students interested in the option pricing applications described in chapters 7, 8, and 9. However, it may be useful to students in Economics. It is also necessary background for the relatively advanced optimization models and methods which are covered in chapters 10, 11, and 12.

- Chapter 7 is a new chapter which is devoted to binomial and trinomial lattices, which were not treated extensively in the first edition. The main issues here are proper implementation and memory management.

- Chapter 8 is naturally linked to chapter 4 and deals with more advanced applications of Monte Carlo and low-discrepancy sequences to exotic options, such as barrier and Asian options. We also deal briefly with the estimation of option sensitivities (the Greeks) by Monte Carlo methods. Emphasis is on European-style options; pricing American options by Monte Carlo methods is a more advanced topic which must be analyzed within an appropriate framework, which is done in chapter 10.

- Chapter 9 applies the background of chapter 5 to option pricing by finite difference methods.

- Chapter 10 deals with numerical dynamic programming. The main reason for including this chapter is pricing American options by Monte Carlo simulation, which was not covered in the first edition but is gaining more and more importance. I have decided to deal with this topic within an appropriate framework, which is dynamic stochastic optimization. In this chapter we just cover the essentials, which means discrete-time and finite-horizon dynamic programs. Nevertheless, we try to offer a reasonably firm understanding of these topics, both because of their importance in Economics and because understanding dynamic programming is helpful in understanding stochastic programming with recourse, which is the subject of the next chapter.

- Chapter 11 deals with linear stochastic programming models with recourse. This is becoming a standard topic for people in Operations Research, whereas people in Economics are much more familiar with dynamic programming. There are good reasons for this state of the matter, but from a methodological point of view I believe that it is very important to compare this approach with dynamic programming; from a practical point of view, stochastic programming has an interesting potential both for dynamic portfolio management and for option hedging in incomplete markets.

- Chapter 12 also deals with the relatively exotic topic of non-convex optimization. The main aim here is introducing mixed-integer programming, which can be used for portfolio management when practically relevant constraints call for the introduction of logical decision variables. We also deal, very shortly, with global optimization, i.e., continuous non-convex optimization, which is important when we leave the comfortable domain of easy optimization problems (i.e., minimizing convex cost functions or maximizing concave utility functions). We also outline heuristic principles such as local search and genetic algorithms. They are useful to integrate simulation and optimization and are often used in computational economics.

- Finally, we offer three appendices on MATLAB, probability and statistics, and AMPL. The appendix on MATLAB should be used by the unfamiliar reader to get herself going, but the best way to learn MATLAB is by trying and using the online help when needed. The appendix on probability and statistics is just a refresher which is offered for the sake of convenience. The third appendix on AMPL is new, and it reflects the increased role of algebraic languages to describe complex optimization models. AMPL is a modeling system offering access to a wide array of optimization solvers. The choice of AMPL is just based on personal taste (and the fact that a demo version is available on the web). In fact, GAMS is probably much more common for economic applications, but the concepts are actually the same. This appendix is only required for chapters 11 and 12.

Finally, there are many more pages in this second edition: more than 600 pages, whereas the first edition had about 400. Actually, I had a choice: either including many more topics, such as interest-rate derivatives, or offering a more extended and improved coverage of what was already included in the first edition. While there is indeed some new material, I preferred the second option. Actually, the original plan of the book included two more chapters on interest-rate derivatives, as many readers complained about this lack in the first edition. While writing this increasingly long second edition, I switched to plan B, and interest-rate derivatives are just outlined in the second chapter to point out their peculiarities with respect to stock options. In fact, when planning this new edition, many reviewers warned that there was little hope to cover interest-rate derivatives thoroughly in a limited amount of pages. They require a deeper understanding of risk-neutral pricing, interest rate modeling, and market practice. I do believe that the many readers interested in this topic can use this book to build a solid basis in numerical methods, which is helpful to tackle the more advanced texts on interest-rate derivatives.

Interest-rate derivatives are not the only significant omission. I could also mention implied lattices and financial econometrics. But since there are excellent books covering those topics and I see this one just as an entry point or a complement, I felt that it was more important to give a concrete understanding of the basics, including some less familiar topics. This is also why I prefer using MATLAB, rather than C++ or Visual Basic. While there is no doubt that C++ has many merits for developing professional code, both in terms of efficiency and object orientation, it is way too complex for newcomers. Furthermore, the heavy burden it places on the reader tends to overshadow the underlying concepts, which are the real subject of the book. Visual Basic would be a very convenient choice: It is widespread, and it does not require yet another license, since it is included in software tools that almost everyone has available. Such a choice would probably increase my royalties as well. Nevertheless, MATLAB code can exploit a wide and reliable library of numerical functions and it is much more compact. To the very least, it can be considered a good language for fast prototyping. These considerations, as well as the introduction of new MATLAB toolboxes aimed at financial applications, are the reasons why I am sticking to my original choice. The increasing number of books using MATLAB seems to confirm that it was a good one.

Acknowledgments. I have received much appreciated feedback and encouragement from readers of the first edition of the book. Some pointed out typos, errors, and inaccuracies. Offering apologies for possible omissions, I would like to thank I-Jung Hsiao, Sandra Hui, Byunggyoo Kim, Scott Lyden, Alexander Reisz, Ayumu Satoh, and Aldo Tagliani.

Supplements. As with the first edition, I plan to keep a web page containing the (hopefully short) list of errata and the (hopefully long) list of supplements, as well as the MATLAB code described in the book. My current URL is:

- `http://staff.polito.it/paolo.brandimarte`

For comments, suggestions, and criticisms, my e-mail address is

- `paolo.brandimarte@polito.it`

One of the many corollaries of Murphy's law says that my URL is going to change shortly after publication of the book. An up-to-date link will be maintained both on Wiley Web page:

- `http://www.wiley.com/mathematics`

and on The MathWorks' web page:

- `http://www.mathworks.com/support/books/`

PAOLO BRANDIMARTE
Turin, March 2006

This Page Intentionally Left Blank

From the Preface to the First Edition

Crossroads are hardly, if ever, points of arrival; but neither are they points of departure. In some sense, crossroads may be disappointing, indeed. You are tired of driving, you are not at home yet, and by Murphy's law there is a far-from-negligible probability of taking the wrong turn. In this book, different paths cross, involving finance, numerical analysis, optimization theory, probability theory, Monte Carlo simulation, and partial differential equations. It is not a point of departure, because although the prerequisites are fairly low, some level of mathematical maturity on the part of the reader is assumed. It is not a point of arrival, as many relevant issues have been omitted, such as hedging exotic options and interest-rate derivatives.

The book stems from lectures I give in a Master's course on numerical methods for finance, aimed at graduate students in Economics, and in an optimization course aimed at students in Industrial Engineering. Hence, this is not a research monograph; it is a textbook for students. On the one hand, students in Economics usually have little background in numerical methods and lack the ability to translate algorithmic concepts into a working program; on the other hand, students in Engineering do not see the potential application of quantitative methods to finance clearly.

Although there is an increasing literature on high-level mathematics applied to financial engineering, and a few books illustrating how cookbook recipes may be applied to a wide variety of problems through use of a spreadsheet, I believe there is some need for an intermediate-level book, both interesting to practitioners and suitable for self-study. I believe that students should:

- Acquire *reasonably* strong foundations in order to appreciate the issues behind the application of numerical methods

- Be able to translate and check ideas quickly in a computational environment

- Gain confidence in their ability to apply methods, even by carrying out the apparently pointless task of using relatively sophisticated tools to pricing a vanilla European option

- Be encouraged to pursue further study by tackling more advanced subjects, from both practical and theoretical perspectives

The material covered in the book has been selected with these aims in mind. Of course, personal tastes are admittedly reflected, and this has something to do with my Operations Research background. I am afraid the book will not please statisticians, as no econometric model is developed; however, there is a wide and excellent literature on those topics, and I tried to come up with a complementary textbook.

The text is interspersed with MATLAB snapshots and pieces of code, to make the material as lively as possible and of immediate use. MATLAB is a flexible high-level computing environment which allows us to implement nontrivial algorithms with a few lines of code. It has also been chosen because of its increasing potential for specific financial applications.

It may be argued that the book is more successful at raising questions than at giving answers. This is a necessary evil, given the space available to cover such a wide array of topics. But if, after reading this book, students will want to read others, my job will have been accomplished. This was meant to be a crossroads, after all.

PS1. Despite all of my effort, the book is likely to contain some errors and typos. I will maintain a list of errata, which will be updated, based on reader feedback. Any comment or suggestion on the book will also be appreciated. My e-mail address is: `paolo.brandimarte@polito.it`.

PS2. The list of errata will be posted on a Web page which will also include additional material and MATLAB programs. The current URL is

- `http://staff.polito.it/paolo.brandimarte`

An up-to-date link will be maintained on Wiley Web page:

- `http://www.wiley.com/mathematics`

PS3. And if (what a shame ...) you are wondering who Commander Straker is, take a look at the following Web sites:

- `http://www.ufoseries.com`
- `http://www.isoshado.org`

PAOLO BRANDIMARTE
Turin, June 2001

Part I

Background

1 Motivation

Common wisdom would probably associate the ideas of numerical methods and number crunching to problems in science and engineering, rather than finance. This intuitive view is contradicted by the relatively large number of books and scientific journals devoted to computational finance; even more so, by the fact that these methods are not confined to academia, but are actually used in real life. As a result, there has been a steady increase in the number of academic programs devoted to quantitative finance, both at Master's and Ph.D. level, and they usually include a course on numerical methods. Furthermore, many people with a quantitative or numerical analysis background have started working in finance, including engineers, mathematicians, and physicists.

Indeed, as the term *financial engineering* may suggest, computational finance is a field where different cultures meet. Hence, a wide array of students and practitioners, with diverse background, will hopefully be interested in a book on numerical methods for finance. On the one hand, this is good news for the author. On the other one, the first difficult task is to get everyone on common ground as far as financial theory and the basics of numerical analysis are concerned; if treatment is too brief, there is a significant risk of losing a considerable subset of readers along the way; if it is too detailed, another subset will be considerably bored. The aim of the first three chapters is to "synchronize" readers with a background in Finance and readers with a scientific background, including students in Engineering, Mathematics, and Physics. In chapter 2, we will give the second subset of readers an overview of concepts in finance, with an emphasis on asset pricing and portfolio management. The first subset of readers will find a reasonably self-contained treatment on classical topics of numerical analysis in chapter 3.

In this introductory chapter we want to give a preview of the problems we will deal with, along with some motivation. The reader who is unfamiliar with some topics just outlined here should not be worried, as they are not taken for granted and will be treated thoroughly in the next chapters. We want to make three points:

1. In financial engineering we need numerical methods (section 1.1).

2. We need sophisticated and user-friendly numerical computing environments, such as MATLAB[1] (section 1.2), even if this does not prevent at all the use of (relatively) low-level languages such as Fortran or C++ or spreadsheets such as Microsoft Excel.

3. Whatever software tool we select, we need a reasonably strong theoretical background, as we must often select among competing methods and many things may go wrong with them (section 1.3).

1.1 NEED FOR NUMERICAL METHODS

Probably, the best-known result in financial engineering is the Black-Scholes formula to price options on stocks.[2] Options are a class of *derivatives*, i.e., financial assets whose value depends on another asset, called the underlying. The underlying can also be a non-financial asset, such as a commodity, or an arbitrary quantity representing a risk factor to someone, such as weather, so that setting up a market to transfer risks makes sense. Options are *contracts* with very specific rules for issuing, trading, and accounting. For instance, a European-style call option on a stock gives the holder the right, but not the obligation, of buying a given stock at a given time (maturity, denoted by T), for a prespecified price (the strike price, denoted by K). Similarly, a put option gives the right to sell the underlying asset at a predetermined strike price. In European-style derivatives, the right specified in the contract can only be exercised at maturity T; in American-style derivatives, one can exercise her right at *any* time *before* T, which in this case plays the role of the expiration date of the option.

In the case of a European-style call option, if the asset price at maturity is $S(T)$, then the payoff is $\max\{S(T) - K, 0\}$. The rationale here is that, under idealized assumptions on financial markets, the option holder could purchase

[1]MATLAB is a registered trademark of The MathWorks, Inc. For more information, see `http://www.mathworks.com`.

[2]The formula was published by Fisher Black and Myron Scholes in 1973. A similar research line had been pursued by Robert Merton, and in fact Scholes and Merton were awarded the Nobel prize in Economics in 1997. By that time, unfortunately, Fisher Black was deceased.

the underlying asset at the prevailing price $S(T)$ and immediately sell it at price K. Clearly, the option holder will do so only if this results in a positive profit. Actually, market imperfections, such as transaction costs or bid–ask spreads, prevent such an idealized trade: even if $S(T)$ is the last quoted price, there is no guarantee that the option holder can actually buy the stock at that price. In the book we will neglect such issues, which are related to the *micro-structure* of financial markets.

If we are at a time instant $t < T$, we would like to assign a value, or a fair price, to the option. However, what we know is only the current price $S(t)$ of the underlying asset, whereas its price $S(T)$ at maturity is not known. If we build some mathematical model for the dynamics of the price $S(t)$ as a function of time, we may regard $S(T)$ as a random variable; hence, the payoff is random as well, and there seems to be no trivial way to price this contract. Let $f(S(t), t)$ be the price of the option at time t if the current price of the underlying asset is $S(t)$; to ease the notation burden we will usually write it as $f(S, t)$. It can be shown that, under suitable assumptions, the value of the contract really depends only on t and S, and it satisfies the following partial differential equation (PDE):

$$\frac{\partial f}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 f}{\partial S^2} + rS\frac{\partial f}{\partial S} - rf = 0, \tag{1.1}$$

where r is the risk-free interest rate, i.e., the rate of interest one can earn by investing her money in a safe account, and σ is a parameter related to the volatility of the price of the underlying asset, which is a risky asset. Typically, we are interested in the current value $f(S_0, 0)$, where $S_0 = S(0)$. Equation (1.1), with the addition of suitable boundary conditions linked to the type of option, may be solved analytically in some cases. For instance, if we denote the cumulative distribution function[3] for the standard normal distribution by $N(z) = \mathrm{P}\{Z \leq z\}$, where Z is a standard normal variable, the price C_0 for a European call option at time $t = 0$ is

$$C_0 = S_0 N(d_1) - Ke^{-rT} N(d_2), \tag{1.2}$$

where

$$\begin{aligned} d_1 &= \frac{\ln(S_0/K) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}, \\ d_2 &= \frac{\ln(S_0/K) + (r - \sigma^2/2)T}{\sigma\sqrt{T}} = d_1 - \sigma\sqrt{T}. \end{aligned}$$

This formula is easy to evaluate, but in general we are not so lucky. The complexity of the PDE or of some additional conditions, which we must impose to fully characterize a specific option, may require numerical methods. We will

[3]See appendix B for a refresher on Probability and Statistics.

cover relatively simple numerical methods for solving PDEs, based on finite differences, in chapter 5, and applications to option pricing will be illustrated in chapter 9. Using finite differences, in turn, may call for the repeated solution of systems of linear equations, which is among the topics of chapter 3 on numerical analysis.

Apart from the obvious computational advantage, analytical formulas are of great importance in gaining insights into how different factors affect option prices. They also allow quick calculation of price sensitivities with respect to such factors, which are relevant for risk management. In the book, we will use analytical formulas quite often in order to validate numerical methods, by comparing the numerical result with the theoretically correct one. This is of no practical value by itself, but it is very instructive. Finally, we will also see that when a complex option cannot be priced analytically, knowing an analytical pricing formula for a related simpler option can be of great value. In option pricing by Monte Carlo simulation (see below), analytical pricing formulas may yield control variates useful to reduce variance in the estimate of price.

Nevertheless, we should note that the distinction between numerical and analytical methods is sometimes a bit blurred. It may happen that analytical formulas are quite complicated. As an example, let us consider the following formula, which we give without much explanation[4]:

$$C_J = \sum_{n=0}^{\infty} \frac{e^{-\lambda T}(\lambda T)^n}{n!} \mathrm{E}\left\{C_{\mathrm{BLS}}\left[S_0 X_n e^{-\lambda \chi T}, T, K, \sigma^2, r\right]\right\}.$$

This is a formula for the price of a European-style call option when price jumps are included in the model. The Black–Scholes model assumes continuous paths for prices, and this formula by Robert Merton generalizes to a model in which jumps occur according to a compound Poisson process. Here $C_{\mathrm{BLS}}(S, T, K, \sigma^2, r)$ is the standard Black–Scholes formula with the usual input arguments; λ is related to the rate of jumps, i.e., the expected number of jumps per unit time; X_n is a random variable related to the size of jumps, and expectation in the formula is with respect to this variable; χ is a number which is also related to the probability distribution of jump sizes. Even without fully understanding this formula, which goes beyond the scope of this introductory book, it is clear that evaluating it is not so trivial and calls for some computational approximation. Nevertheless, it gives an explicit representation of the effect of each factor affecting price, whereas in a purely numerical approach this important information is lost.

Even in the simple case of equation (1.2), some numerical method is actually applied, since we have to evaluate the function:

$$N(z) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{z} e^{-y^2/2}\, dy,$$

[4]See [5, page 320] for details.

where the integral cannot be solved in closed form. Here, we may evaluate the integral by quite efficient ad hoc approximation formulas, rather than by general-purpose methods for numerical integration. Sometimes, however, we have to compute or approximate integrals in multiple dimensions. In fact, thanks to a result known as Feynman–Kač formula, the solution of a PDE such as (1.1) can be expressed as an expected value. This and other pricing arguments imply that option prices may be expressed as expected values, which boil down to an integral. Unfortunately, when expectation is taken with respect to *many* random variables, standard methods to compute integrals in low-dimensional spaces fail.

In other problem settings, we have to approximate a *function* defined by an integral. For instance, consider a function $g(x, y)$ and define a function of x by

$$F(x) = \int_a^b g(x, y) f_Y(y)\, dy.$$

Such a situation occurs often in stochastic optimization, when x is a decision variable influencing the result, which is only partially under our control because of the effect of a random "disturbance" Y, whose density is $f_Y(y)$ over the support $[a, b]$ (possibly $(-\infty, +\infty)$). The function $F(x)$ can be considered as the expected cost or profit resulting from our decisions. We will see concrete examples in chapters 10 and 11.

Since computing integrals is so important, chapter 4 is entirely devoted to this topic. Apart from deterministic integration methods, we will also deal extensively with random sampling methods known as Monte Carlo integration or Monte Carlo simulation. Monte Carlo simulation has a incredibly wide array of applications, including option pricing and risk management. For instance, it can be shown that the price of a European call option at time $t = 0$ is given by the following expected value:

$$C = \mathrm{E}^{\mathbb{Q}}\left[e^{-rT} \max\{S_T - K, 0\}\right],$$

where S_T is the (random) price of the underlying asset at maturity, and the expected value is taken under a suitably chosen probability measure (denoted by $\mathbb{Q}$). In other words, the option value is the expected value of the payoff, discounted back to time $t = 0$, under a certain probability measure. If we are able to generate M independent random samples $S_T^{(j)}$, $j = 1, \ldots, M$, of the asset price, under probability measure $\mathbb{Q}$, then by the law of large numbers we could estimate the expected value by the sample mean

$$\hat{C} = \frac{1}{M} \sum_{j=1}^{M} e^{-rT} \max\{S_T^{(j)} - K, 0\}.$$

This is the essence of Monte Carlo simulation, and a number of tricks of the trade are needed in order to obtain a reliable and computationally efficient estimate.[5] Variance reduction methods and alternative integration approaches based on low-discrepancy sequences will be introduced in chapter 4, and applications to option pricing are illustrated in chapter 8.

Another widely applied approach to option pricing is based on binomial or trinomial lattices. These can be regarded as a sort of clever discretization of the underlying stochastic process. From this point of view, they are a deterministic way to generate sample paths, whereas Monte Carlo is based on random sample path generation. Another point of view is that certain finite difference approaches can be regarded as generalization of trinomial lattices. We will see applications of these methods in chapter 7.

Another major topic of the book is optimization, which is introduced in chapter 6. Optimization models and methods play many different roles in finance. In the option pricing context, optimization is at the core of pricing American-style options. Since American-style options may be exercised at any time before expiration, optimal exercise strategies must be accounted for in pricing. For instance, in an American-style call option, it would be tempting to exercise the option as soon as it gets *in-the-money*, i.e., when $S(t) > K$ for a call option and you could earn an immediate profit. However, one should also wonder if it could be better to wait for a better opportunity. This is not a trivial problem; indeed, it can be shown that it is *never* optimal to exercise an American-style call option on a stock, unless it pays dividends before expiration.

An older type of application of optimization methods is portfolio management. Given a set of assets in which one can invest her wealth, we must decide how much should be allocated to each one of them, given some characterization of the uncertainty in assets return. The best-known portfolio optimization model is based on the idea of minimizing the variance of portfolio return (a measure of risk), while meeting a constraint on its expected value. This leads to mean-variance portfolio theory, a topic pioneered by Harry Markowitz in the 1950s. While somewhat idealized, this model had an enormous practical and theoretical impact, eventually earning Markowitz a Nobel prize in Economics in 1990.[6] Since then, many different approaches to portfolio optimization have been developed, and they will be illustrated in chapters 10, 11, and 12.

[5]As we mentioned, option pricing by solving a partial differential equation or by computing an expectation are theoretically equivalent approaches, via Feynman–Kač formula. However, they can be quite different in computational terms. It is interesting to note that, historically, Black–Scholes formula was first obtained by solving the pricing PDE analytically, whereas the recent tendency is to use expectation based approaches because of their generality.

[6]Markowitz shared the prize with Merton Miller and William Sharpe. What is probably less known is that he was among the developers of SimScript, one of the first programming languages for discrete-event simulation. By the way, Robert Merton had a background in engineering. This shows how artificial the barriers between Economics and Engineering may be.

It is also important to note that asset pricing and portfolio optimization are not necessarily disjoint topics. Many Financial Economics theories are based on portfolio optimization models which in turn lead to asset pricing models. We will not cover these topics, however, both because of space limitations and because they are not strictly related to numerical methods.

There are still other kinds of application of optimization methods, which may more instrumental, such as parameter fitting or model calibration. In complex markets, asset prices may depend on a set of unobservable parameters, and one would like to introduce and price a new asset, in a way which is coherent with observed prices for other traded assets. To do so, a typical approach is the following. First we build a theoretical pricing model, depending on such parameters. Then we try to find values for these parameters, which are as coherent as possible with observed prices. Let $\boldsymbol{\alpha}$ be the vector of unknown parameters; according to the asset pricing model, the theoretical price of asset j should be $\hat{P}_j(\boldsymbol{\alpha})$, whereas the observed price is P_j^o. We would like to get a vector of parameters yielding the best fit. A standard way to do so is solving the following optimization model:

$$\min_{\boldsymbol{\alpha}} \sum_j \left(P_j^o - \hat{P}_j(\boldsymbol{\alpha}) \right)^2 .$$

Then, given the optimal set of parameters, we may proceed to price new assets using the theoretical model. This type of approach is essential in pricing interest-rate derivatives. Interest-rate derivatives are considerably more difficult to analyze than options on stocks and are outside the scope of this book; we will just outline the related issues in section 2.8.

As expected, some simple optimization models may be solved analytically, yielding quite useful insights. However, as a rule, very sophisticated computational approaches are needed.

1.2 NEED FOR NUMERICAL COMPUTING ENVIRONMENTS: WHY MATLAB?

MATLAB is an interactive computing environment, providing both basic and sophisticated functions. You may use built-in functions to solve possibly complex but standard problems, or you may devise your own programs by writing them as M-files, i.e., as text files including sequences of instructions written in a high-level matrix-oriented language. Moreover, MATLAB has a rich set of graphical capabilities, which we will use in a very limited fashion, including the ability of quickly developing graphical user interfaces. The unfamiliar reader is referred to appendix A for a quick tour of MATLAB programming.

Some classical numerical problems are readily solved by MATLAB functions. They include:

- Solving systems of linear equations

- Solving non-linear equations in a single unknown variable (including polynomial equations as a special case)
- Finding minima and maxima of functions of a single variable
- Approximating and interpolating functions
- Computing definite integrals (in low-dimensional spaces)
- Solving ordinary differential equations, as well as some simple PDEs

This and much more is included in the basic MATLAB core. More complex versions of these problems may be solved by other MATLAB ready-to-use functions, but you have to get the appropriate toolbox. A toolbox is simply a set of functions written in the MATLAB language, and it is usually provided in source form, so that the user may customize or use the code as a starting point for further work. For instance, the Optimization toolbox is needed to solve complex optimization problems, involving several decision variables and possibly complex constrains, as well as to solve systems of non-linear equations. Another relevant toolbox for finance is the Statistics toolbox, which includes many more functions than we will use. In particular, it offers functions to generate pseudorandom numbers that are needed to carry out Monte Carlo simulations. Based on the Statistics and Optimization toolboxes, a Financial toolbox was first devised a few years ago, which included different groups of functionalities. Some were low-level functions aimed at date and calendar manipulation or finance-oriented charting, which are building blocks for real-life applications; others dealt with simple fixed-income assets, portfolio optimization, and derivatives pricing.

After this first toolbox, others were introduced which are directly related to finance:

- GARCH toolbox
- Financial time series toolbox[7]
- Financial derivatives toolbox
- Fixed-income toolbox

We will not deal with such toolboxes in the book, but information can be obtained by browsing The MathWorks' Web site (`http://www.mathworks.com`). We should also mention that other toolboxes, which were not specifically developed for financial applications, could be useful, such as the PDEs toolbox

[7]At the time of writing, the functionalities of this toolbox have been included in the Financial toolbox.

or the genetic and direct search toolbox.[8] Other more instrumental tools are useful to develop professional applications, such as Excel link, Web server, the compiler, or the Datafeed module enabling web connections to different financial web sites.

Now the question is: Why choose MATLAB for this book? Indeed, there are different competitors, at different levels:

- User-friendly spreadsheets, such as Microsoft Excel. In fact, there are spreadsheet-based books showing how optimization and simulation methods may be applied to financial problems. Spreadsheets are equipped with solvers able to cope with small-scale mathematical programming problems, and extensions are available to run Monte Carlo simulations or optimization by genetic algorithms.

- On the opposite side of the spectrum, one could use low-level languages such as C++ or Fortran. C++ seem a favorite, if you look at the number of books on computational finance based on this language, but there are people maintaining that the recent versions of Fortran do still have some advantages. C++ or Fortran may be used either to implement the algorithms directly or to call available scientific computing libraries.

- There are also specialized libraries or environments, such as statistical or optimization tools.

How does MATLAB compare against such alternatives? The obvious answer is that the choice is largely a matter of taste, and it depends on your aim.

Sure, when you have to carry out simple computations, there's little point in resorting to a full-fledged computing environment, and probably spreadsheets are the best choice. However, the extra effort in learning a programming language pays off when you have to program a complex numerical method which goes beyond what is standard and readily available. Actually, there is no way to really learn numerical methods without some knowledge of a programming language, and in any case, even if you use a spreadsheet as the front end, it is quite likely that you have to write some code in Visual Basic or C++.

Compiled languages such as Fortran and C++ are certainly the most efficient option, in terms of execution speed.[9] If you have to write really lightning-fast code, this is the best choice.

[8]Genetic algorithms and direct search methods are optimization methods which do not require computing derivatives of the objective function. This makes them very flexible for some types of optimization models, as we will see in chapters devoted to optimization.
[9]A compiled language is based on the translation of source level code to machine level language. You need a compiler to do that; optimized compilers are able to obtain extremely fast code. An interpreter does not translate to machine level code, but to some internal form which is then executed. Usually an interpreter has some advantage in terms of debugging and flexibility, which is paid in terms of execution speed.

MATLAB is an interpreted language, and even if it is quite efficient, there is some difference. However, the performance gap is being bridged by increasingly fast MATLAB versions. Furthermore, executable libraries can be generated from MATLAB code by using the MATLAB compiler; these libraries can then be linked within the application just as any C++ code. But the most important advantage of MATLAB is that it is a very simple, yet powerful, programming language. Unlike C++, you may avoid bothering with issues such as memory allocation, variable declaration, etc. MATLAB is an excellent rapid prototyping tool: You may implement a quite complex algorithm with a very limited amount of lines. Simple code means less time to develop and less chances for programming bugs. Then, if it is really needed, you may go on by translating the prototyped code to, e.g., C++. This is obviously important in a practical setting, but it is not really essential in a didactic book like the present one. When learning a numerical method, being distracted by too many programming details is certainly bad.

MATLAB can be thought of as a suitable compromise between conflicting requirements. The increasing number of toolboxes and books using MATLAB is a good proof of that. Needless to say, this does not imply that MATLAB has no definite limitations. When one has to deal with large-scale optimization problems, it is necessary to resort to specialized packages such as CPLEX,[10] against which MATLAB is unlikely to be competitive (it should be noted that the Optimization toolbox is aimed at general non-linear programming, whereas some optimization packages deal only with linear and quadratic programming). Furthermore, mixed-integer programming problems[11] cannot be solved, at present, by MATLAB.[12] Even worse, when you have a large optimization model, loading the data in a form suitable to a numerical library function is a difficult and error-prone task without the support of algebraic modeling languages such as AMPL.[13] This is one of the reasons why, in the chapters on optimization models, we will sometimes solve them using AMPL. This should not place any burden on the reader, since a free demo version can be downloaded from the AMPL web site. See appendix C for a quick tour of AMPL.

By the same token, if one is interested in statistical computing applied to finance, it is quite likely than some of the many econometric packages are

[10]CPLEX is a registered trademark of ILOG. See `http://www.ilog.com`.

[11]Mixed-integer programming models are optimization models in which some decision variables are restricted to integer, rather than real, values. They are dealt with in chapter 12. See also example 1.2 on page 15.

[12]We should mention that the latest release of the Optimization Toolbox does include a solver for certain pure binary (0/1) linear programming. However, this is not suitable to large scale mixed-integer programming.

[13]AMPL (A Mathematical Programming Language) was originally developed at Bell Laboratories. At present it is available in many versions through different sellers, including ILOG, under license from the copyright owner. See `http://www.ampl.com`.

better suited to the task. The point is that none of these offers the many functionalities of MATLAB within a single integrated environment.

To summarize, we may argue that a product like MATLAB is the best *single* tool to lay down good foundations in numerical methods. Cheap MATLAB student editions are available, and its use in finance is spreading. So we believe that learning MATLAB is definitely an asset for students and practitioners in financial engineering.

A last choice had to be made in writing the book: To which extent should toolboxes be used? On the one hand, using too many toolboxes would place some burden on the reader, who may not have access to all of them. On the other hand, using only the MATLAB core would probably limit what we can do. So, again, a compromise must be reached. Our choice has been to use a very limited subset of functions from the Statistical and Financial toolbox, which can be easily replicated. We will sometimes use functions from the Optimization toolbox, but the same results can be obtained by the free AMPL demo version. We will use neither advanced financial toolboxes nor the Partial Differential Equations Toolbox. This choice is somewhat contradictory: Why use the Optimization toolbox and not the PDEs one? The point is that there is a wide gap between a conceptual statement of optimization methods, and a robust working implementation. It is not the aim of this book to bridge that gap, so we will avoid a detailed treatment of most optimization methods, limiting ourselves to the principles behind them. On the contrary, simple finite difference methods are relatively easy to implement, and can be treated in detail. Finally, we should also note that typical computational finance courses do cover basic finite difference methods for solving PDEs, but not sophisticated optimization methods.

1.3 NEED FOR THEORY

Now that we established that we are going to use MATLAB in the book, another question may arise: Why should we bother learning numerical *methods*, when they are already available in professionally crafted, ready-to-use code? Can we get rid of theory? Although, in most cases, there is no need for a deep knowledge of numerical analysis in order to use MATLAB, there are at least three reasons to gain a basic understanding of the theoretical background of numerical methods.

1. Without a sound background, you cannot go on developing your own solutions when the available methods are not enough.

2. Without a sound background, you cannot choose the most appropriate algorithm when alternatives are given.

3. Without a sound background, you cannot use methods properly and, most important, you cannot understand what is going wrong when results are not reasonable or you get weird error messages.

In particular, we need some understanding of fundamental issues like "conditioning of a numerical problem" and "stability of an algorithm." These concepts are briefly discussed in chapter 3. Here we give some simple examples of the trouble one can get into without a sound knowledge of the pitfalls of numerical computing.

Example 1.1 Consider the following expression:

$$9 \cdot 8.1 + 8.1$$

Everyone would agree that this is just a complicated way to write $10 \times 8.1 = 81$. Let us try it on a computer, using MATLAB:

```
>> 9 * 8.1 + 8.1
ans =
   81.0000
```

Everything seems right. Now, there is a built-in function in MATLAB, `fix`, which can be used to round a number to the integer nearest to zero.[14] Note that `fix` does *not* round to the nearest integer:

```
>> fix(4.1)
ans =
     4
>> fix(4.9)
ans =
     4
```

Let us try it on the expression above:

```
>> fix(9*8.1 + 8.1)
ans =
    80
```

Now something seems quite wrong. Actually, the point is that the first result is not what it looks like. This may be seen by changing the visualization format of numbers and trying again:

```
>> format long
>> 9 * 8.1 + 8.1
ans =
  80.99999999999999
```

Actually, there was some warning, since MATLAB printed `81.0000` rather than `81`, as it happens with

[14]The reader is urged to explore the differences between `fix` and the related functions `floor`, `ceil`, and `round`.