ELSEVIER FINANCE

computational finance

numerical methods for pricing financial instruments

George Levy

CD ROM INCLUDED

COMPUTATIONAL FINANCE

COMPUTATIONAL FINANCE

Numerical Methods for Pricing Financial Instruments

George Levy

ELSEVIER
BUTTERWORTH
HEINEMANN

AMSTERDAM BOSTON HEIDELBERG LONDON NEW YORK OXFORD
PARIS SAN DIEGO SAN FRANCISCO SINGAPORE SYDNEY TOKYO

Butterworth-Heinemann
Elsevier
Linacre House, Jordan Hill, Oxford OX2 8DP
200 Wheeler Road, Burlington, MA 01803

First published 2004

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

Library of Congress Cataloguing in Publication Data
A catalogue record for this book is available from the Library of Congress

ISBN 0 7506 5722 7

Typeset by Integra Software Services Pvt. Ltd, Pondicherry, India
www.integra-india.com
Printed and bound in The Netherlands

To Kathryn

Contents

Preface

It was in late 1995 to early 1996 (shortly after the birth of his first daughter Claire) that the author first began to read the currently available finance books in order to write C/C++ financial software. However, apart from the book *Options Futures and Other Derivatives* by John Hull, he found very little information of practical help and had to trawl through the original journal articles in the Bodleian library for more information. Even then much information on how to implement and test various models was not included.

The current book aims to provide practical information on basic computational finance. In addition many statistical, financial, and numerical results are derived so that the reader does not need to consult a large number of other books. It should be mentioned that many of the code excerpts assume that the reader has access to NAG Ltd numerical libraries. However, for those who are not so fortunate, equivalent C/C++ software is provided on the accompanying CD ROM.

The book is divided into three parts. Part I considers the type of interfaces to financial functions that can be created using the Microsoft Windows environment. In particular it deals with the use of Dynamic Link Libraries (DLLs) and ActiveX components from languages such as Visual Basic, VBScript, VB.NET, and C#. The author considers that one of the main developments in technical computing over the past ten years has been the emergence of technologies that permit the rapid development of easy to use interfaces to complex functions. At the mouse click of a virtual button complicated computations can be performed.

Part II of the book is concerned with the mathematics of option pricing, and covers computational methods for vanilla options and also simple barrier options. In many cases more exotic options (that for example include complex barriers, lockout periods, rebates, etc.) can be created from these by using them as building blocks. Most of this material can be understood using basic college mathematics and its presentational style is inspired by Numerical Recipies, for instance see Press *et al.* (1992).

Finally Part III of the book deals with financial econometrics and the modelling of volatility. Although the main emphasis is on GARCH, Levy processes, and stochastic volatility models are also considered.

From an historical point of view the finite-difference methods used in Part II have their origin in the numerical weather forecasting techniques proposed by Lewis Richardson between 1910 and 1930, see Richardson (1910) and Richardson and Gaunt (1927). These were later developed by Phyliss Nicolson (Girton College Cambridge) and John Crank in the 1940s, and their method is known as the

Crank–Nicolson finite-difference method. GARCH time series methods can trace their roots to earlier work in the 1920s concerned with AR processes. We could continue by discussing the history of Gaussian processes, Levy distributions, etc. However, the reader can read about this elsewhere.

It should be mentioned that this is not a book about how to use and trade in various financial derivatives. In fact the author does not have this experience, and books such as John Hull are a good introduction to this subject.

I would like to take this opportunity to thank my wife Kathryn for putting up with the extra time that a book such as this requires.

I would also like to thank the series editor, Dr Steven Satchell, for his very useful advice concerning the structure of the book, and Mike Cash of Butterworth-Heinemann for his support throughout the project.

In addition I gratefully acknowledge the Risk Waters Group for allowing PDF versions of several journal articles to be placed on the CD ROM.

George Levy
Benson 2003

Part I

Using Numerical Software Components within Microsoft Windows

Chapter 1

Introduction

This part of the book describes a variety of Microsoft technologies that enable software developers to deploy their numerical/financial functions within Microsoft Windows. It would be impossible in such a short space to provide a comprehensive description of Microsoft Windows. One of the reasons is that Microsoft regularly brings to market new and improved products. For instance in 2002 Microsoft launched its release version of .NET; this had been previously available in the form of Beta 1 and Beta 2 releases. This product includes the languages VB.NET, an updated version of Visual Basic, and C#. The main purpose of .NET is to facilitate the easy deployment of *Web Service* component software over the Internet. Currently (October 2002) the full MSDN documentation and help system (with information on .NET) takes well over 1 Gbyte of computer disc space. Voluminous books have also been written on various aspects of .NET such as: VB.NET, C#, XML, and XSL, and these can be consulted as required. Here we can only aim at providing a short introduction to the use of Microsoft technology for numerical computation. In order to combat information overload we will try here to convey the maximum *essential* information in the minimum space. To achieve this we will adopt the strategy of supplying well commented code excerpts from real (working) Microsoft projects. It is intended that these code excerpts can be used as templates for the creation of computational finance components. Additional material, including documentation, complete source code and ready to use Microsoft projects can be found on the CD ROM which accompanies this book.

Before embarking on a more detailed description of various Microsoft languages and applications it would be sensible to try and gain an overview of the Microsoft Windows environment and consider the possible benefits to be gained from using it for software development.

To a large extent the Microsoft Windows environment is all about the Visual user-interface. The replacement of command line, DOS based, programming by Microsoft Windows heralded an explosion in the use of computers. Esoteric DOS commands (understood by only a few) gave way to the simple interactive user-interface. Here the user can control a program by (for example) clicking Windows buttons with the mouse and entering values into Windows textboxes. The enormous advantage of this approach (now used by nearly all computational software) is that the user is shielded from complicating factors such as the operating system and the underlying computer languages. All the user needs to do is to enter the correct data and click the appropriate button; the answer then appears on the screen.

Using Windows software can now be made as easy as turning on the television or playing a video player. However, as with the *real button* on the television or video

player remote control, the *virtual button* of a Windows application can conceal a great deal of underlying technology. The purpose of this part of the book is to provide information concerning the type of Windows software that may be invoked when a Windows *event* (such as a mouse click) occurs.

We will consider the ways in which numerical and financial *components* can be incorporated into various Windows applications. Here we take the term numerical and financial *component* to mean a self-contained computational object which, given certain inputs, will return various computed results. The inputs and computed results can be single values (*scalars*), one-dimensional arrays (*vectors*), two-dimensional arrays (*matrices*), or higher dimensional arrays. The components described here are designed to be used in *mixed language* applications. This means that the component is created using a computationally efficient language such as C/C++ or Fortran, and resides in either a Windows *Dynamic Link Library* (DLL) or *COM ActiveX Control*. It is then used from another (interface) language such as Visual Basic, which *wraps* it and provides the Visual interactive interface seen by the user. If the components are to be accessible from the complete range of Microsoft languages it is good programming practice to restrict their data types to the very basic C/C++ types such as `real`, `double`, and `long` (Fortran types `REAL`, `DOUBLE PRECISION`, and `INTEGER`) which have equivalents in all the other Microsoft languages. It should be noted that, in C++, seemingly innocent structures, strings and character parameters can be particularly difficult (if not impossible) to deal with.

The topics covered here include:

- DLL creation using Visual C++.
- Calling C and Fortran routines from Visual Basic, VB.NET, and C#.
- Using ActiveX and COM components from Visual Basic, Internet Web pages, Excel, and Delphi.
- Scripting ActiveX components on Internet Web pages using VBScript and JScript.
- XML and transformation using XSL.

The section on XML data representation and transformation was included because it provides an introduction to viewing data (or computed results) with the Web browser Internet Explorer 6. In Chapter 6 we show how the use of XSL style sheets permits an XML file to be transformed into a HTML file. This tranformation can be accomplished automatically when the XML file is loaded into a Web browser (for example by double clicking the XML file with a mouse). By using different XSL files it is thus possible to obtain different views of the numeric values contained within an XML file. For example it may be considered appropriate to generate both a *tabular view* which gives columns of numeric values, and also a *report view* which contains fewer numbers and contains graphical plots that summarize the information.

Information is given on how to call components from Visual Basic, Delphi, VB.NET, and C#. In addition we show how numeric components can be used from within Windows applications such as Excel and Internet Explorer.

As previously mentioned we will not consider in any detail the construction of the Visual interface; this information can be readily found in the large selection of Microsoft Windows books that are currently available. We will also concentrate on

the mixed language use of numeric components. This means that although all the examples in this part of the book could have been written in Visual C++, they use a variety of Windows languages such as Visual Basic, VBScript, Delphi, etc.

In practical terms this means that the creation of a computational finance application is a two-step process:

- The creation of the numerical/finance component, using a computationally efficient language such as Visual C++ or Visual Fortran.
- The construction of the application framework and user-interface using Microsoft languages such as Visual Basic, VB.NET, C#, etc.

This separation leads to a natural division of labour. The numerical components are created by an expert mathematician/numerical analyst (with limited knowledge of languages such as C++, Visual Basic, etc.) and the construction of the Visual interface is performed by a computer programmer (with limited numerical knowledge) who is expert in the more complex features of the language chosen for developing the application's visual interface. For example a numerical analyst may create an option pricing component using Visual C++. A computer programmer may then incorporate this component into a variety of applications such as: Web-based services using VB.NET or C#, spreadsheet applications using Excel, or stand-alone PC applications using Visual Basic, Delphi, etc.

Finally here are just a few remarks concerning the style of the book.

Small example applications have been included in the areas of statistics, linear algebra, financial derivative pricing, portfolio optimisation, and numerical optimisation.

Also some of the examples refer to the NAG C library DLL and also the NAG Fortran DLL. However, the techniques used in these examples can easily be applied to calling functions from other, user-defined, Windows DLLs.

Care has been taken to make all the computer code as simple as possible. We don't (intentionally) try to be clever; the main consideration is that the code works. Readers can always modify the code to suit their needs and preferences.

Finally some people may find the style rather terse compared to the coverage given in other books. This is intentional, since there is so much the information presented will be limited to the minimum required to obtain working software. The book has been written from the author's experience that:

A page of working (and well commented) computer code is worth a hundred pages of explanation.

In spite of all these caveats it is hoped the reader will find the information in the following sections both instructional and useful reference material.