About the Companion C++ Code

This book comes with a professional implementation in C++ freely available to readers on:

www.wiley.com/go/computationalfinance

In this repository, readers will find:

1. All the source code listed or referenced in this publication.
2. The files AAD*.*[1] constitute a self contained, general-purpose AAD library. The code builds on the advanced techniques exposed in this publication, in particular those of Chapters 10, 14, and 15, to produce a particularly fast differentiation library applicable to many contexts. The code is progressively built and explained in Part III.
3. The files mc*.*[2] form a generic, parallel, financial simulation library. The code and its theoretical foundations are described in Part II.
4. Various files with support code for memory management, interpolation, or concurrent data structures, such as threadPool.h, which is developed in Part I and used throughout the book to execute tasks in parallel.
5. A file main.h that lists all the higher level functions that provide an entry point into the combined library.
6. A Visual Studio 2017 project wrapping all the source files, with project settings correctly set for maximum optimization. The code uses some C++ 17 constructs, so the project setting "C++ Language Standard" on the project property "C/C++ / Language / C++ Language Standard" must be set to "ISO C++ 17 standard." This set-

ting is correctly set on the project file xlComp.vcxproj, but readers who compile the files by other means must be aware of this.

7. A number of xl*.* files that contain utilities and wrappers to export the main functions to Excel, as a particularly convenient front end for the library. The project file xlComp.vcxproj is set to build an xll, a file that is opened from Excel and makes the exported library functions callable from Excel like its standard functions. We wrote a tutorial that explains how to export C++ code to Excel. The tutorial ExportingCpp2xl.pdf is available in the folder xlCpp along with the necessary source files. The wrapper xlExport.cpp file in our project precisely follows the directives of the tutorial and readers can inspect it to better understand these techniques.
8. Finally, we provide a pre-built xlComp.xll[3] and a spreadsheet xlTest.xlsx that demonstrates the main functions of the library. All the figures and numerical results in this publication were obtained with this spreadsheet and this xll, so readers can reproduce them immediately. The computation times were measured on an iMac Pro (Xeon W 2140B, 8 cores, 3.20 GHz, 4.20 max) running Windows 10. We also carefully checked that we have *consistent* calculation times on a recent quad core laptop (Surface Book 2, i7-8650U, 4 cores, 1.90 GHz, 4.20 max), that is, (virtually) identical time in single threaded mode, twice the time in multi-threaded mode.

The code is entirely written in standard C++, and compiles on Visual Studio 2017 out of the box, without any dependency to on a third-party library.

NOTES

1 With a dependency on gaussians.h for the analytic differentiation of Gaussian functions, and blocklist.h for memory management, both included.

2 With dependency on various utility files, all included in the project.

3 To run xlComp.xll, readers may need to install Visual Studio redistributables VC_redist.x86.exe and VC_redist.x64.exe, also included in the repository.