Introduction

In the aftermath of the global financial crisis of 2008, massive regulations were imposed on investment banks, forcing them to conduct frequent, heavy regulatory calculations. While these regulations made it harder for banks to conduct derivatives businesses, they also contributed to a new golden age of computational finance.

A typical example of regulatory calculation is the CVA (Counterparty Value Adjustment),[1] an estimation of the loss subsequent to a future default of a counterparty when the value of the sum of all transactions against that counterparty (called netting set) is positive, and, therefore, lost. The CVA is actually the value of a real option a bank gives away whenever it trades with a defaultable counterparty. This option is a put on the netting set, contingent on default. It is an exotic option, and a particularly complex one, since the underlying asset is the netting set, consisting itself in thousands of transactions, some of which may be themselves optional or exotic. In addition, the netting set typically includes derivatives transactions in different currencies on various underlying assets belonging to different asset classes. Options on a set of heterogeneous underlying assets are known to the derivatives industry and called hybrid options. Investment banks' quantitative research departments actively developed hybrid models and related numerical implementations in the decade 1998–2008 for the risk management of very profitable transactions like Callable Reverse Power Duals (CRPD) in Japan.

The specification, calibration, and simulation of hybrid models are essentially well known; see, for example, [7], [8], and [9]. What is unprecedented is the vast number of cash flows and the high dimension of the simulation. With a naive implementation, the CVA on a large netting set can take minutes or even hours to calculate.

In addition, the market and credit risk of the CVA must be hedged. The CVA is a cost that impacts the revenue and balance sheet of the bank, and its value may change by hundreds of millions when financial and credit markets move. In order to hedge the CVA, it is not enough to compute it. All its sensitivities to market variables must also be produced. And a CVA is typically sensitive to thousands of market variables: all the underlying assets that affect the netting set, all the rate and spread curves and volatility surfaces for all the currencies involved, as well as all the foreign exchange rates and their volatility surfaces, and, of course, all the credit curves. In order to compute all these risks with traditional finite differences, the valuation of the CVA must be repeated thousands of times with inputs bumped one by one.

This is of course not viable, so investment banks first implemented crude approximations, at the expense of accuracy, and distributed calculations over large data centres, incurring massive hardware, development, and maintenance costs.

Calculation speed became a question of survival, and banks had to find new methodologies and paradigms, at the junction of mathematics, numerics, and computer science, in order to conduct CVA and other regulatory calculations accurately, practically, and quickly on light hardware.

That search for speed produced new, superior mathematical modeling of CVA (see, for instance, [10]), a renewed interest in the central technology of scripting derivatives cash flows (see our dedicated volume [11]), the systematic incorporation of parallel computing (Part I of this volume), and exciting new technologies such as Algorithmic Adjoint Differentiation (AAD, Part III of this volume) that computes thousands of derivatives sensitivities in constant time.

In the early 2010s, head of Danske Markets Jens Peter Neergaard was having lunch in New York with quantitative research legend Leif Andersen. As Leif was complaining about the administration and cost of data centres, JP replied:

We calculate our CVA on an iPad mini.

In the years that followed, the quantitative research department of Danske Bank, under Jesper Andreasen's leadership, turned that provocative statement into reality, by leveraging cutting-edge technologies in a software efficient enough to conduct CVA risk on light hardware without loss of accuracy.

In 2015, at a public Global Derivatives event in Amsterdam, we demonstrated the computation of various xVA and capital charge over a sizeable netting set, together with all risk sensitivities, within a minute on an Apple laptop. That same year, Danske Bank won the In-House System of the Year Risk award.

We have also been actively educating the market with frequent talks, workshops, and lectures. These publications are the sum of that work.

Modern quantitative researchers must venture beyond mathematics and numerical methods. They are also expert C++ programmers, able to leverage modern hardware to produce highly efficient software, and master key technologies like algorithmic adjoint differentiation and scripting. It is the purpose of our publications to teach these skills and technologies.

It follows that this book, like the other two volumes in the series, is a new kind of publication in quantitative finance. It constantly combines financial modeling, mathematics, and programming, and correspondences between the three, to resolve real-life financial problems and improve the accuracy and performance of financial derivatives software. These publications are inseparable from the professional source code in C++ that comes with them. The publications build the libraries step by step and the code demonstrates the practical application of the concepts discussed in the books.

Another unique aspect of these publications is that they are not about models. The definitive reference on models was already written by Andersen and Piterbarg [6]. The technologies described in our publications: parallel simulations, algorithmic adjoint differentiation, scripting of cash-flows, regression proxies, model hierarchies, and how to bring them all together to better risk manage derivatives and xVA, are all model agnostic: they are designed to work with all models. We develop a version of Dupire's popular model [12] for demonstration, but models have little screen time. The stars of the show are the general methodologies that allow the model, any model, to compute and differentiate on an iPad mini.

This volume, written by Antoine Savine, focuses on algorithmic adjoint differentiation (AAD) (Part III), parallel programming in C++ (Part I), and parallel simulation libraries (Part II). It is intended as a one-stop learning and reference resource for AAD and parallel simulations, and is complete with a professional implementation in C++, freely available to readers in our source repository.

AAD is a ground-breaking programming technique that allows one to produce derivative sensitivities to calculation code, automatically, analytically, and most importantly *in constant time*. AAD is applied in many scientific fields, including, but not limited to, machine learning (where it is known under the name “backpropagation”) or meteorology (the powerful improvement of expression templates, covered in Chapter 15, was first suggested by a professor of meteorology, Robin Hogan). While a recent addition, AAD has quickly become an essential part of quantitative finance and an indispensable centrepiece of modern financial libraries. It is, however, still misunderstood to a large extent by a majority of finance professionals.

This publication offers a complete coverage of AAD, from its theoretical foundations to the most elaborate constructs of its efficient implementation. This book follows a pedagogical logic, progressively building intuition and taking the time to explain concepts and techniques in depth. It is also a complete reference for AAD and its application in the context of (serial and parallel) financial algorithms. With the exception of Chapters 12 and 13,[2] Part III covers AAD in itself, without reference to financial applications. Part III and the bundled AAD library in C++ can be applied in various contexts, including machine learning, although it was tested for maximum performance in the context of parallel financial simulations.

A second volume [11], co-authored by Jesper Andreasen and Antoine Savine, is dedicated to the scripting of derivatives cash flows. This central technology is covered in detail, beyond its typical usage as a convenience for the structuring of exotics. Scripts are introduced as a practical, transparent, and effective means to represent and manipulate transactions and cash flows in modern derivatives systems. The publication covers the development of scripting in C++, and its application in finance, to its full potential. It is also complete with a professional implementation in C++. Some advanced extensions are covered, such as the automation of fuzzy logic to stabilize risks, and the aggregation, compression, and decoration of cash flows for the purpose of xVA.[3]

A third (upcoming) volume, written by Jesper Andreasen, Brian Huge, and Antoine Savine, explores effective algorithms for the computation and differentiation of xVA, and covers the details of the efficient implementation, applications, and differentiation of the LSM[4] algorithm.

NOTES

1 We have a lot of similar regulatory calculations, collectively known as xVA. The capital charge for derivatives businesses and the cost of that capital are also part of these calculations.

2 As well as part of 14.

3 We briefly introduced CVA. The other value adjustments banks calculate for funding, liquidity, cost of capital, etc. are collectively known as xVA and the computation techniques detailed for CVA are applicable with some adjustments.

4 Least Squares Method, sometimes also called the Longstaff-Schwartz Model, invented in the late 1990s by Carriere [13] and Longstaff-Schwartz [14], and briefly introduced in Section 5.1.