Conclusion

After an introduction to parallel computing, and the construction of a parallel simulation library, we explored in detail the theory and implementation of AAD. As a result, we could compute accurate market and model risk sensitivities over parallel simulations in fractions of a second.

We don't believe that such results could be achieved with a free or commercial AAD library applied as a black box over instrumented code. This kind of speed is only obtained with an intimate understanding of AAD and its implementation, combined with a deep expertise in the differentiated algorithms. Selective instrumentation, a clever pathwise back-propagation, and a systematic application of check-pointing are the key ingredients, along with an efficient AAD library, boosted with expression templates, and an efficient instrumented simulation code. That is why we covered both AAD and financial simulations in this publication. We also provided computationally efficient C++ code, one that is not meant to be used as a black box, but instead provides a flexible interface for expert users to accommodate many types of designs and applications. (Readers who made it to the conclusion are, by definition, expert users.)

While AAD over parallel simulations is central to effective, modern derivatives systems, it is not by itself sufficient for the efficient calculation and differentiation of complex derivatives books and xVA adjustments.

Other key technologies are at play there. In particular, we covered in detail the mathematical representation of financial products, includ-

ing xVA, and their cash-flows, in Chapter 4, but we did not implement it in code, delivering instead a few sketchy examples for the production of numerical results. The representation and manipulation of transactions, down to cash-flows, in a consistent, modular, efficient, transparent, and flexible form, constitutes another keystone of modern computational finance, and the subject of a dedicated publication [11], co-authored by Jesper Andreasen.

Other key notions that we discussed but did not develop in code include regression proxies and model hierarchies. These are covered in a third (upcoming) volume, co-authored by Jesper Andreasen and Brian Huge. The third volume will also cover in detail the application of all those technologies to regulatory calculations like xVA, and the specific algorithms to calculate and differentiate of xVA *on an iPad mini*.