Calculating risk in real time

Luca Capriotti, *Jacky Lee* and *Matthew Peacock* describe a new technique that can hugely reduce the time taken to calculate credit risk in a Monte Carlo simulation

For any number of underlying assets or names in a portfolio, adjoint algorithmic differentiation allows the calculation of the full first order risk at a computational cost which is at most four times the cost of calculating the profit and loss statement of the portfolio itself. This opens the way to real time risk management for the most challenging applications such as the hedging of counterparty credit risk.

Managing counterparty credit risk

Setting up a theoretically sound framework for managing counterparty credit risk is challenging because it requires the evaluation of all the trades facing a given counterparty within a consistent pricing methodology. For multi-asset portfolios this typically comes with extraordinary computational challenges, and a high infrastructure cost.

For a given portfolio of trades facing the same investor or institution, the credit valuation adjustment (CVA) aims to capture the expected loss associated with the counterparty defaulting in a situation in which the position, netted for any collateral agreement, has a positive mark-to-market for the dealer. This can be evaluated at time $T_0 = 0$ as

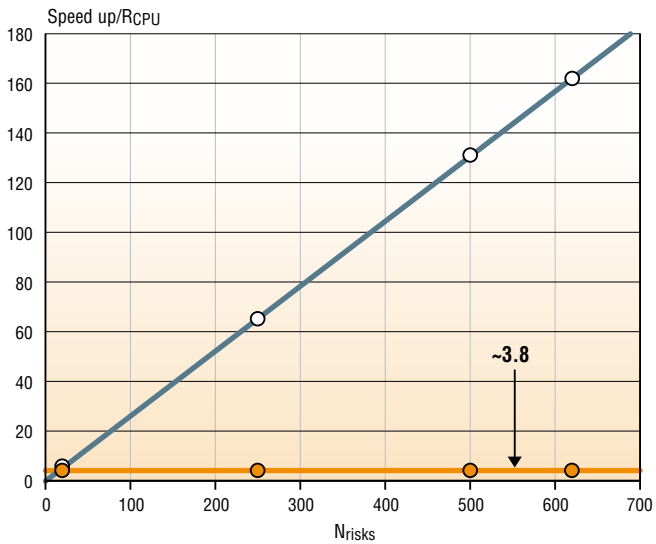
$$V_{\rm CVA} = \mathbb{E}\Big[\mathbb{I}(\tau_c \le T) D(0,\tau_c) \times L_{\rm GD}(\tau_c)\Big(NPV(\tau_c) - C(R(\tau_c^-))\Big)^+\Big]$$

where τ_c is the default time of the counterparty, $NPV(t)$ is the net present value of the portfolio at time t from the dealer's point of view, $C(R(t))$ is the collateral outstanding, typically dependent on the rating R of the counterparty, $L_{\rm GD}(t)$ is the loss given default, $D(0,t)$ is the discount factor for the interval $[0,t]$, and $\mathbb{I}(\tau_c \le T)$ is the indicator that the counterparty's default happens before the longest deal maturity in the portfolio, T.

Here for simplicity of notation we consider the unilateral CVA; the generalisation to bilateral CVA is straightforward. In general, the quantity above depends on several correlated random market factors, including interest rate, counterparty's default time and rating, recovery amount, and all the market factors that the net present value of the portfolio depends on. As such, its calculation requires a Monte Carlo simulation.

Standard approaches for the calculation of risk require repeating the calculation of the profit and loss statement (P&L) of the portfolio under hundreds of market scenarios. As a result, in many cases these calculations cannot be completed in a practical amount of time, even employing a vast amount of computer power. Since the total cost of the through-the-life risk management can determine whether it is profitable to execute a new trade, solving this technology problem is critical to allow a securities firm to remain competitive.

Speed of CVA calculation



Speed-up in the calculation of risk for the CVA of a portfolio of five commodity swaps over a five-year horizon, as a function of the number of risks computed (empty dots). The full dots are the ratio of the computer time required for the calculation of the CVA, and its sensitivities, and the computer time spent for the computation of the CVA alone. Lines are guides for the eye.

Speeding up the risk calculation

Algorithmic differentiation (AD) is a set of programming techniques for the efficient calculation of the derivatives of functions implemented as computer programs. The main idea underlying AD is that any such function – no matter how complicated – can be interpreted as a composition of basic arithmetic and intrinsic operations that are easy to differentiate.

What makes AD particularly attractive, when compared to standard (finite-difference) methods for the calculation of derivatives, is its computational efficiency.

In fact, AD exploits the information on the structure of the computer code in order to optimise the calculation. In particular, when one requires the derivatives of a small number of outputs with respect to a large number of inputs, the calculation can be highly optimised by applying the chain rule through the instructions of the program in opposite order with respect to their original evaluation. This gives rise to the adjoint (mode of) algorithmic differentiation (AAD).

The main ideas underlying AAD can be illustrated by considering a function

$$\theta \to P$$

mapping a real vector θ to a real scalar P through a sequence of steps

$$\theta \to \ldots \to V \to \ldots \to P$$

Here, each step can be a distinct high-level function or even an individual instruction.

The adjoint mode of AD results from propagating the derivatives of the final result with respect to all the intermediate variables – the so called *adjoints* – until the derivatives with respect to the independent variables are formed. Using the standard AD notation, the adjoint of any intermediate variable V_k is defined as

$$\bar{V}_k = \bar{Y} \frac{\partial Y}{\partial V_k}$$

where $\bar{Y}$ is a constant (that we can think for instance equal to 1). Starting from the adjoint of the outputs, $\bar{Y}$, by applying the chain rule we can calculate the adjoint of the intermediate variables working from right to left

$$\bar{\theta} \leftarrow \ldots \leftarrow \bar{V} \leftarrow \ldots \leftarrow \bar{P}$$

until we obtain $\bar{\theta}$, i.e.,

$$\bar{\theta}_k = \bar{P} \frac{\partial P}{\partial \theta_k}$$

the gradient of the function $\theta \to P$ multiplied by $\bar{P}$.

One particularly important theoretical result is that the execution time of such adjoint recursion is bounded by approximately four times the cost of execution of the original one, irrespective of the number of inputs. Thus, one can obtain the sensitivity of a single output to an unlimited number of inputs for a little more work than the original calculation.

In the context of a Monte Carlo simulation, these ideas can be used for a highly efficient implementation of the so-called path-wise derivative method.

Testing the adjoint algorithmic differentiation calculation

As a numerical test we present here results for the calculation of risk on the CVA of a portfolio of swaps on commodity futures. In this case, the remarkable computational efficiency of the AAD implementation is clearly illustrated in the chart, left. Here we plot the speedup produced by AAD with respect to the standard finite-differences method. On a fairly typical trade horizon of five years, for a portfolio of five swaps referencing distinct commodities futures with monthly expiries, the CVA bears nontrivial risk to over 600 parameters: 300 futures prices, and at-the-money volatilities, (say) 10 points on the zero rate curve, and 10 points on the CDS curve of the counterparty used to calibrate the transition probabilities of the rating transition model.

As illustrated in the chart, the computer time required for the calculation of the CVA, and its sensitivities, is less than four times the time spent for the computation of the CVA alone, as predicted by the general result quoted above. As a result, even for this very simple application, AAD produces risk over 150 times faster than finite differences; that is, for a CVA evaluation taking 10 seconds, AAD produces the full set of sensitivities in less than 40 seconds, while finite differences require approximately 1 hour and 40 minutes.

Moreover, as a result of the analytic integration of the singularities introduced by the rating process, the risk produced by AAD is typically substantially less noisy than the one produced by finite differences, so that a dramatically reduced number of iterations is required to achieve a satisfactory Monte Carlo confidence interval, thus resulting in an additional remarkable speedup.

In conclusion, adjoint algorithmic differentiation allows an extremely efficient calculation of counterparty credit risk valuations in Monte Carlo. The scope of this technique is clearly not limited to this important application but extends to virtually any valuation performed either with stochastic or deterministic numerical methods.

The opinions and views expressed in this paper are uniquely those of the authors, and do not necessarily represent those of Credit Suisse Group.

References and notes for this piece may be found online.


Luca Capriotti


Jacky Lee


Matthew Peacock

Luca Capriotti is a director, Jacky Lee is a managing director and Matthew Peacock is a vice-president in quantitative strategies, at Credit Suisse. All three work in New York in the global credit product quant strats team, of which Lee is global head and Capriotti is US regional head. The authors hold PhDs in theoretical physics (from the International School of Advanced Studies), operations research (Stanford) and engineering (Sydney), respectively.