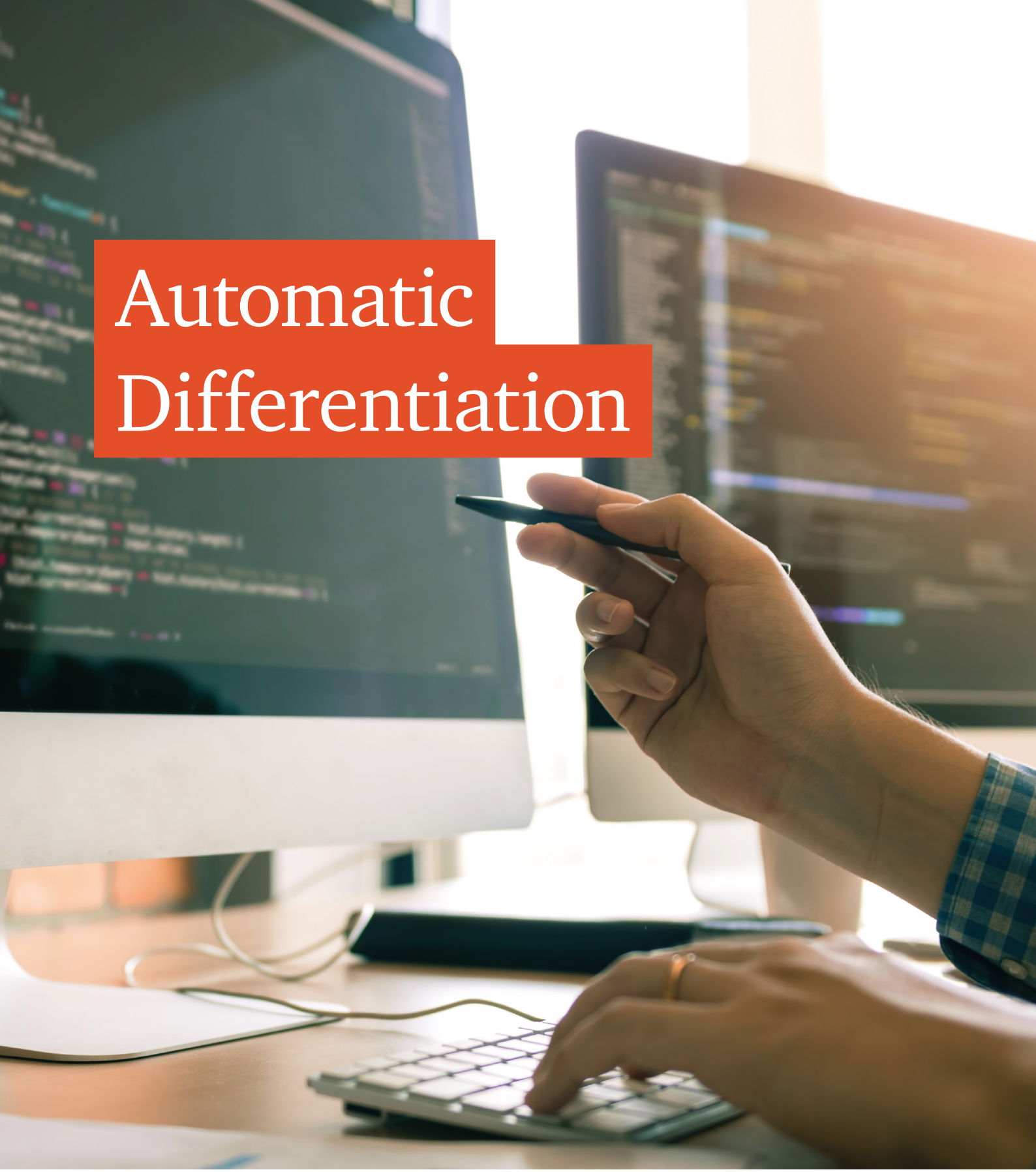
Automatic Differentiation



pwc

Automatic Differentiation

This article serves as a summary of the main concepts regarding Automatic Differentiation (AutoDiff) and its application in the context of CVA. Automatic Differentiation is a family of techniques for evaluating derivatives of numeric functions expressed as computer programs with unprecedented speed and accuracy. We accompany our theoretical discussion with the results of a numerical experiment where we compute CVA sensitivities using Automatic Differentiation and the standard method finite differences, and demonstrate that AutoDiff finishes the computations up to hundreds of times faster.

1 Introduction and outline

AutoDiff is ubiquitous in the field of machine learning, specifically for gradient-based optimisation procedures (see Baydin et al. (2018) for an excellent survey), but also has a broad spectrum of applications in the world of finance, specifically in risk management, hedging, calibration and computation of regulatory capital. Here we refer to the works of Kaebe et al. (2009), Capriotti (2011) and Henrard (2013). Specifically for the application of AutoDiff in relation to XVAs, we mention the works of Capriotti et al. (2011) and Green (2015). We also highlight the implicit application of AutoDiff in the context of finance through recent developments in the application of neural networks in the area of hedging by Buehler et al. (2019), the solving of optimal stopping problems by Becker et al. (2019) and the calibration of local stochastic volatility models using generative adversarial networks (GANs) by Cuchiero et al. (2020) to mention a few. One generally differentiates between two modes of Automatic Differentiation, namely between the *Forward mode* and the *Reverse mode*. The Reverse mode is generally preferred for functions which have a large number of inputs and a small number of outputs, whereas the opposite holds true for the Forward mode. In this work, we only discuss the Reverse mode of Automatic Differentiation, having finance applications in mind. Therefore, for the purposes of this work, when we refer to Automatic Differentiation or AutoDiff we mean Reverse-mode Automatic Differentiation. Moreover, we only discuss the case where the objective function is real-valued and depends on multiple inputs, and note that our discussion can easily be generalised to the case where the objective function has multiple outputs. For an in-depth introduction to both the Forward and Reverse mode of Automatic Differentiation we refer to Griewank et al. (2008). The rest of this article is structured as follows. In Section 2, we formally outline the problem setting and give a brief overview of the finite difference method for the numerical calculation of derivatives. In Section 3, we discuss the main concepts behind Automatic Differentiation and the motivations behind its use. Afterwards, in Section 4, we give an introduction to CVA and how AutoDiff can be applied in this context. Section 5 presents the results of the numerical experiment, where we have compared the AutoDiff runtime to the finite difference runtime for CVA sensitivity calculations. Finally, we conclude our discussion and results in Section 6.

2 Problem setting and finite differences

Let $X \subset \mathbb{R}^n$ be open for some $n \in \mathbb{N}$. By $C^1(X)$ we denote the set of continuously differentiable functions $f : X \to \mathbb{R}$. We fix an $f \in C^1(X)$ throughout this section and the next section. The object of interest is the set of partial derivatives $(\partial f(x)/\partial x^1, ..., \partial f(x)/\partial x^n)$, i.e. the gradient of f with respect to a given $x \in X$. A classical approach to compute the latter the is *finite differences* method. Let us define $x(i, \varepsilon) = (x^1, ..., x^i + \varepsilon, ..., x^n)$, for $i = 1, ..., n$, where $\varepsilon > 0$ and small enough such that $x(i, \varepsilon) \in X$, for all $i = 1, ...n$. Note that the existence of such an ε is guaranteed since X is open. The finite differences method approximates the gradient of f with respect to x by means of

$$\frac{\partial f(x)}{\partial x^i} \simeq \frac{f(x(i, \varepsilon)) - f(x)}{\varepsilon}, \ i = 1, ..., n.$$

Applying this procedure to obtain the gradient of f requires $n + 1$ evaluations of the objective function f, which makes the application of finite differences a very intensive computational task, in the case of a high-dimensional input space X, which is usually the case in the field of finance or machine learning. Moreover, the choice of ε always comes with a tradeoff between accuracy and numerical stability. We refer to Baydin et al. (2018) for further details on the finite differences method and its comparison to AutoDiff.

3 Automatic Differentiation

AutoDiff uses the idea of “divide and conquer”, where the function f is represented as a composition of simpler

functions for which the derivatives are straighforward to calculate, and the final result is aggregated by means of the chain rule. In the sequel we follow a similar description of Automatic Differentiation as provided by Griewank et al. (2008) and Capriotti (2011). AutoDiff consists of two steps, namely the *forward pass* and the *backward pass*. In the forward pass, the function $x \longmapsto f$ is evaluated at the input x by passing it through a sequence of elementary operations. These elementary operations can be represented by means of scalar intermediate variables $w^1, ..., w^n, w^{n+1}, ..., w^k$, where the first n variables are used to initialize x, i.e. $w^1 = x^1, ..., w^n = x^n$, and we have $w^\ell = f^\ell(\{w^j\}_{j \prec \ell})$ for $\ell = n+1, ..., k$, where $\{w^j\}_{j \prec \ell}$ indicates the set of all w^j, $1 \leq j < \ell$, such that w^ℓ depends explicitly on w^j and f^ℓ is an elementary scalar function. We notice that such representations are not unique and that during the forward pass the values of the intermediate variables and their dependencies are stored. Usually, such a sequence of operations can be represented as a computational graph which we visualise schematically as

$$x = (w^1, ..., w^n) \rightarrow w^\ell = f^\ell(\{w^j\}_{j \prec \ell}) \quad \text{for } \ell = n+1, ..., k-1 \ \rightarrow w^k = y,$$

where we have written $f(x) = y$ for convenience. Next, we describe the backward pass. For this purpose, we define the adjoints $\overline{w^\ell}$ as $\partial y / \partial w^\ell$ for $\ell = 1, ..., k$. We notice that $\overline{w^k} = 1$ holds by definition. Note that we can then write

$$\overline{w^\ell} = \frac{\partial y}{\partial w^\ell} = \sum_{j \succ \ell} \frac{\partial y}{\partial w^j} \frac{\partial w^j}{\partial w^\ell} = \sum_{j \succ \ell} \overline{w^j} \frac{\partial w^j}{\partial w^\ell},$$

for $\ell = 1, ..., k-1$, where we have used the chain rule in the second equality and the sum runs over $j > \ell$ such that w^j depends explicitly on w^ℓ. Therefore, we actually get the object of interest $\overline{x^i} = \overline{w^i} = \partial y / \partial w^i$, for $i = 1, ..., n$, by backward propagation of the adjoints, i.e.

$$\overline{x} = (\overline{w^1}, ..., \overline{w^n}) \leftarrow \overline{w^\ell} \text{ for } \ell = k-1, ..., n+1 \leftarrow \overline{w^k} = 1,$$

which leverages the information obtained in the forward pass.

We end this section by borrowing a theoretical result from Griewank et al. (2008) which is known as the *cheap gradient principle*, and demonstrates the efficiency of AutoDiff. Informally, it states

"The computational cost of evaluating the derivatives of a scalar objective function via Reverse-mode Automatic Differentiation is at most 4 times the computational cost of evaluating the objective function."

This makes Reverse-mode Automatic Differentiation an appealing tool for finance and machine learning applications where the objective function is real-valued but the input space X is usually high-dimensional. Alongside the efficiency, an additional attractive attribute of AutoDiff is the accuracy with which the derivatives are computed. No approximations are needed as in the finite difference method. AutoDiff performs all computations in an exact manner (up to machine precision).

4 Applications in the world of CVA

Credit Valuation Adjustment (CVA) is the adjustment to the fair value of derivative instruments in order to take into account the downgrade of the counterparty quality. Formally, let $(\Omega, \mathcal{F}, \mathbb{F}, \mathrm{Q})$ be a filtered probability space, where $\mathbb{F} = (\mathcal{F}_t)_{t \geq 0}$ is a filtration assumed to satisfy the usual conditions, and all processes are assumed to be adapted to $\mathbb{F}$. The CVA is defined by

$$\mathfrak{C} = E_{\mathrm{Q}} \left[(1 - R) V_\tau^+ 1_{\tau \leq T} \right],$$

where $E_{\mathrm{Q}}[\cdot]$ denotes the expectation under Q, R represents the recovery rate and is $[0, 1)$-valued, τ is an $\mathbb{F}$-stopping time which can be interpreted as the default time of the counterparty, and $V = (V_t)_{t \geq 0}$ denotes the process which describes the discounted mark-to-market value of the portfolio the bank holds with the counterparty subject to netting and collateral agreements. Moreover, we defined $x^+ = \max(0, x)$, $x \in \mathbb{R}$. For $\mathfrak{C}$ to be well-defined, we also assume that V_t is Q-integrable for all $t \geq 0$. Furthermore, T denotes the longest maturity of a contract in the portfolio that the bank has with the counterparty.

We introduce the $\mathbb{R}^n$-valued risk factor process $Z = (Z_t^1, ..., Z_t^n)_{t \geq 0}$, which we assume is driving the values of the transactions the bank has with the counterparty and the credit rating of the counterparty, hence V and τ depend on Z. The propagation of the risk factors Z is usually governed by a stochastic differential equation, with given initial values $Z_0^1, ..., Z_0^n$, which are either directly observable or market implied. This makes $\mathfrak{C}$ actually a function of these initial values, i.e. $\mathfrak{C} \equiv \mathfrak{C}(Z_0^1, ..., Z_0^n)$. The object of interest is then $\mathfrak{X}$, the set of sensitivities of the CVA with respect to the initial values of the risk factors, i.e.

$$\mathfrak{X} = \left(\frac{\partial \mathfrak{C}}{\partial Z_0^1}, ..., \frac{\partial \mathfrak{C}}{\partial Z_0^n} \right).$$

The set of sensitivities $\mathfrak{X}$ is used for risk management and hedging purposes (see Capriotti et al. (2011), Ruiz (2015) and Green (2015)). Moreover, under MAR50 (see BCBS (2020)), $\mathfrak{X}$ can be used for the computation of CVA regulatory capital under the SA-CVA framework (which is less conservative in terms of regulatory capital than the BA-CVA frameowork, which does not require the computation of $\mathfrak{X}$). For all the applications mentioned, speed and accuracy play a detrimental role, both of which can be achieved by AutoDiff. We also mention at this point that it is only

rarely possible to evaluate $\mathfrak{C}$ analytically but instead one has to rely on Monte Carlo techniques, where the time horizon $[0,T]$ is discretised and replaced by a set $\{0 = t_0 < t_1 < ... < t_N = T\}$, the risk factor process Z is simulated on this discretised time grid and the expectation $E_{\mathbb{Q}}[\cdot]$ is replaced by the empirical mean[1].

To illustrate the above points, we discuss a specific example. In BCBS (2011), $\mathfrak{C}$ is approximated by $\widetilde{\mathfrak{C}}$, which is defined as

$$\widetilde{\mathfrak{C}} = (1-R)\sum_{i=0}^{N-1}\frac{E_{\mathbb{Q}}[V_{t_{i+1}}^{+}] + E_{\mathbb{Q}}[V_{t_i}^{+}]}{2}\Delta P_{t_i}^{+},$$

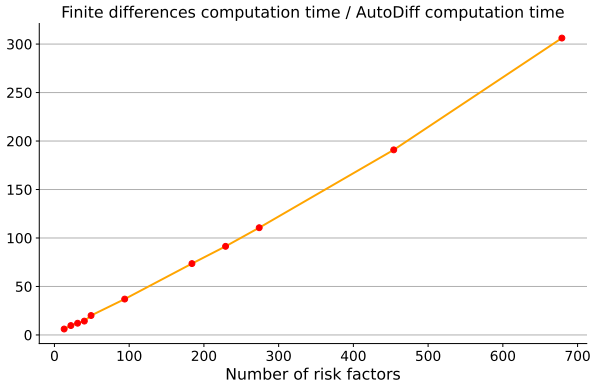
where, ΔP_{t_i} is given by $P_{t_i} - P_{t_{i+1}}$ and $P_{t_i} = -\exp(s_{t_i}\cdot t_i/(1-R))$, for $i = 0, ..., N$, and s_t denotes the counterparty CDS spread with maturity t, which can be obtained from a finite set of market-observable CDS spreads, for example by a cubic C^2 spline interpolation. We assume that V_{t_i} can be written as $V_{t_i} = G_{t_i}(Z_{t_i})$, for some function $G_{t_i} : \mathbb{R}^n \to \mathbb{R}$, for $i = 0, ..., N$, and that Z is a Markov process which is discretised using a simple Euler scheme such that the discrete-time dynamics of Z are governed by $Z_{t_{i+1}} = F_{t_i}(Z_{t_i}, W_i)$, for $i = 0, ..., N-1$, where $F_{t_i} : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}^n$ and W_i is an $\mathbb{R}^m$-valued random variable. For a number of M i.i.d. simulations of the random variables $W_0, ..., W_i$, which we denote by $W_{0,j}, ..., W_{i,j}$, for $j = 1, ..., M$, we can approximate $E_{\mathbb{Q}}[V_{t_{i+1}}^{+}]$ by

$$E_{\mathbb{Q}}[V_{t_{i+1}}^{+}] \simeq \frac{1}{M}\sum_{j=1}^{M}(G_{t_{i+1}}(F_{t_i}(...F_{t_1}(Z_0, W_{0,j})..., W_{i,j})))^{+}.$$

The previous approximation makes it evident that $E_{\mathbb{Q}}[V_t^{+}]$ is a function of $Z_0 = (Z_0^1, ..., Z_0^n)$. Combining the previous observations, we can see that $\mathfrak{C}$ is a function of the initial market observable CDS spreads of the counterparty, as well as the initial values of the risk factors. The applicability of AutoDiff for this specific example boils down to the regularity properties of the interpolation scheme of the counterparty CDS spreads, as well of the functions G_{t_i} and F_{t_i}.

5 A numerical experiment

In order to demonstrate the efficiency of Automatic Differentiation, we have conducted a numerical experiment for calculating the CVA sensitivities of portfolios consisting of the same number of equity options and interest rate caps, the latter having either monthly or quarterly payment frequencies. All trades are assumed to be part of a single netting set and to have a maturity of 1Y. For equity options, the value is determined by the Black-Scholes formula, whereas the relevant equity risk factors are simulated in a Heston model. For the material mentioned we refer to Hilpisch (2015). Interest rate caps are valued using the analytical formula available in the Hull-White 1 factor model, and the short rate is also simulated using the latter (see Brigo et al. (2006)). A recovery rate $R = 0$ together with a flat discount curve equal to 1 is assumed and a weekly time grid is chosen to simulate the risk factors. In addition to that, all trades and model parameters are subject to random initialisation[2] subject to certain conditions in order for the trades and simulations to make economic sense, and the CDS spreads and interest rate yields are interpolated using a cubic C^2 spline. The CVA is calculated by means of $\widetilde{\mathfrak{C}}$ from above, and the CVA sensitivities $\mathfrak{X}$ are calculated with respect to equity spot prices, equity volatilities, interest rate volatilites as well as interest rate yields and CDS spreads of the counterparty across pre-specified tenors. We denote by $t_{FD}(n)$ or $t_{AD}(n)$ the amount of time the finite differences method or Automatic Differentiation needed for computing the CVA sensitivities with respect to n risk factors. The figure below depicts $t_{FD}(n)/t_{AD}(n)$ as a function of n, and can be interpreted as the number of times Automatic Differentiation is faster than finite differences for a given number of CVA sensitivity calculations. The red dots represent actual calculations with respect to the risk factors mentioned above for a given number of trades, whereas the orange line represents a linear interpolation between the red dots and is present only for visual purposes.



As is evident from the figure, Automatic Differentiation enables CVA sensitivities to be calculated faster by factors than the standard method finite differences. For example, if the calculation of CVA sensitivities with respect to 700 risk factors would take 1.5 minutes using Automatic Differentiation, the corresponding calculation using finite differences would take around 7.5 hours.

[1] The applicability of AutoDiff in this context depends on the dynamics of the risk factor process Z, as well as the relationship between the risk factors and the discounted value of the portfolio V and the default time τ. We note, however, that the assumptions needed for the applicability of AutoDiff are usually satisfied in practical applications, and refer the reader to Glasserman (2004) and Protter (1990) for the technical details.

[2] Neither client data, nor PwC data were used in the numerical experiments.

6 Conclusion

In this article we have outlined the main principles regarding the evaluation of derivatives of numeric functions. We have briefly described the main concepts regarding the Reverse mode of Automatic Differentiation and presented an example application in the context of CVA. Specifically, we have demonstrated how AutoDiff can be used for the efficient calculation of CVA sensitivities, which in turn can be used for effective risk management, hedging, as well as for the calculation of regulatory capital. We have demonstrated in our numerical experiments that AutoDiff can perform calculations hundreds of times faster than the commonly used finite differences method. The speed and accuracy provided by AutoDiff open the gates for a next-level risk management. Importantly, we note that the scope of applications of AutoDiff is not isolated to the area of CVA, but extends in general to applications where sensitivity based calculations need to be performed.

References

Baydin, A.G., Pearlmutter, B.A., Radul, A.A., and Siskind, J.M., *Automatic differentiation in machine learning: a survey*, Journal of Marchine Learning Research 18: 1-43, 2018.

BCBS, *Basel III: A global regulatory framework for more resilient banks and banking systems*, `https://www.bis.org/publ/bcbs189.pdf`, 2011.

BCBS, *MAR50 - Credit Valuation Adjustment Framework*, `https://www.bis.org/basel_framework/chapter/MAR/50.htm?inforce=20230101&published=20200708`, 2020.

Becker, S., Cheridito, P., and Jentzen, A., *Deep optimal stopping*, Journal of Machine Learning Research 20: 1-25, 2019.

Brigo, D., and Mercurio, F., *Interest rate models-Theory and Practice*, Springer-Verlag, Berlin, 2006.

Buehler, H., Gonon, L., Teichmann, J., and Wood, B., *Deep hedging*, Quantitative Finance 19 (8): 1271-1291, 2019.

Capriotti, L., *Fast Greeks by algorithmic differentiation*, The Journal of Computational Finance 14 (3): 3-35, 2011.

Capriotti, L., Lee, J., and Peacock, M., *Real time counterparty credit risk management in Monte Carlo*, https://ssrn.com/abstract=1824864, 2011.

Cuchiero, C., Khosrawi, W., and Teichmann, J., *A generative adversarial network approach to calibration of local stochastic volatility models*, Risks 8 (4): 101, 2020.

Giles, M., and Glasserman, P., *Smoking adjoints: Fast monte carlo greeks*, Risk 19 (1): 88-92, 2006.

Glasserman, P., *Monte Carlo methods in financial engineering*, Springer-Verlag, New York, 2004.

Green, A., *XVA: credit, funding and capital valuation adjustments*, John Wiley & Sons, 2015.

Griewank, A., and Walther, A., *Evaluating derivatives: principles and techniques of algorithmic differentiation*, Society for industrial and applied mathematics, 2008.

Henrard, M., *Calibration in finance: Very fast greeks through algorithmic differentiation and implicit function*, Procedia Computer Science 18: 1145-1154, 2013.

Hilpisch, Y., *Derivatives analytics with Python: data analysis, models, simulation, calibration and hedging*, John Wiley & Sons, 2015.

Kaebe, C., Maruhn, J.H., and Sachs, E.W., *Adjoint-based Monte Carlo calibration of financial market models*, Finance and Stochastics 13 (3): 351-379, 2009.

Ruiz, I., *Xva desks: A new era for risk management*, Palgrave Macmillan, London, 2015.

Protter, P., *Stochastic integration and differential equations*, Spinger-Verlag, Berlin, 1990.

Contacts

Andrin Bernet
Partner, Financial Risk Management,
PwC Switzerland
+41 58 792 24 44
andrin.bernet@pwc.ch

Manuel Plattner
Director, Financial Risk Management,
PwC Switzerland
+41 58 792 14 82
manuel.plattner@pwc.ch

Patrick Mijatovic
Senior Associate, Financial Risk Management,
PwC Switzerland
+41 58 792 40 92
patrick.mijatovic@pwc.ch