XVA Principles, Nested Monte Carlo Strategies, and GPU Optimizations

Lokman A. Abbas-Turki, Stéphane Crépey, Babacar Diallo

XVA Principles, Nested Monte Carlo Strategies, and GPU Optimizations

Lokman A. Abbas-Turki[1], **Stéphane Crépey**[2], **Babacar Diallo**[1,2,3]
[1] Laboratoire de Probabilités et Modèles Aléatoires, UMR 7599, Université Pierre-et-Marie Curie
[2] LaMME, Univ. Evry, CNRS, Université Paris-Saclay, 91037, Evry, France
[3] Quantitative Research GMD/GMT Crédit Agricole CIB, 92160 Montrouge

We present a nested Monte Carlo (NMC) approach implemented on graphics processing units (GPU) to X-valuation adjustments (XVA), where X ranges over C for credit, F for funding, M for margin, and K for capital. The overall XVA suite involves five compound layers of dependence. Higher layers are launched first and trigger nested simulations on-the-fly whenever required in order to compute an item from a lower layer. If the user is only interested in some of the XVA components, then only the sub-tree corresponding to the most outer XVA needs be processed computationally. Inner layers only need a square root number of simulation with respect to the most outer layer. Some of the layers exhibit a smaller variance. As a result, with GPUs at least, error controlled NMC XVA computations are doable. But, although NMC is naively suited to parallelization, a GPU implementation of NMC XVA computations requires various optimizations. This is illustrated on XVA computations involving equities, interest rate, and credit derivatives, for both bilateral and central clearing XVA metrics.

Keywords: X-Valuation Adjustment (XVA), Nested Monte Carlo (NMC), Graphics Processing Units (GPU).

1. Introduction

Since the 2008 financial crisis, investment banks charge to their clients, in the form of rebates with respect to the counterparty-risk-free value of financial derivatives, various add-ons meant to account for counterparty risk and its capital and funding implications. These add-ons are dubbed XVAs, where VA stands for valuation adjustment and X is a catch-all letter to be replaced by C for credit, D for debt, F for funding, M for margin, or K for capital.

XVAs greatly complicate the derivative pricing equations by making them global, nonlinear, and entity-dependent. However, in order to avoid or defer major IT changes, many banks (but not all) tend to content themselves with the following exposure-based approach (see e.g. Cesari, Aquilina, Charpillon, Filipovic, Lee, and Manda (2010)): First, they compute the mark-to-market cube of the counterparty-risk free valuation of all their contracts in any scenario and future time point. Then they integrate in time the ensuing expected positive exposure (EPE) profile against the hazard function of the bank implied from its CDS curve. A similar approach is applied to DVA and FVA computations.

Unquestionably, exposure profiles, whether considered in expectation, such as with the EPE, or at some higher quantile levels, with the potential future exposure

(PFE) involved in the determination of regulatory trading limits, are important.

However, an exposure-based approach is purely static, whereas a dynamic perspective is required for (even partial) XVA hedging purposes and for properly accounting for the feedback effects between different XVAs (e.g. from the CVA into the FVA). Moreover, an exposure-based XVA approach essentially assumes independence between the market and credit sides of the problem: Beyond more or less elaborate patches such as the ones proposed in Pykhtin (2012), Hull and White (2012), Li and Mercurio (2015), or Iben Taarit (2017), it is hard to extend rigorously to wrong-way risk, which is the risk of adverse dependence between the credit risk of the counterparty and the underlying market exposure. Last but not least, an exposure-based XVA approach comes without error control, at least whenever the exposure is computed by global regression as done in certain banks. This is due to the unconditional approximations involved in such global regressions.

Instead, in this paper, we explore a full simulation, nested Monte Carlo XVA computational approach granting a $\mathrm{O}(M_{(0)}^{-\frac{1}{2}})$ mean square error, where $M_{(0)}$ is the outer number of trajectories, optimally implemented on GPUs.

Albanese, Bellaj, Gimonet, and Pietronero (2011) advocate a supercomputer XVA implementation, whereby risk factors are simulated forward, whereas the backward pricing task is performed by fast matrix exponentiation in floating arithmetics. Although extremely fast and accurate whenever applicable, this approach is restricted to models written as piecewise time homogeneous Markov chains (for applicability of the matrix exponentiation formula) with at most three factors (unless advanced techniques are used for circumventing the curse of dimensionality), which may not be an option in all banks. Malliavin calculus can be used for extending such an approach to more standard (space continuous) models (see Abbas-Turki, Bouselmi, and Mikou (2014)), but the curse of dimensionality issue remains.

The **outline** of the paper is as follows. Section 2 sets the XVA stage. Section 3 presents the multi-layered dependence between the different XVA metrics and discusses their NMC implementation from a high-level perspective. Section 4 illustrates the above by three case studies. Section 5 discusses future perspectives. Sections A through D detail the related GPU programming optimization techniques.

As we see it, the **main contributions** of this work are:

- From a high-level perspective, the conceptual view of Section 3 on the dependence between the XVA metrics, and its algorithmic counterpart in the form of the nested Monte Carlo Algorithm 1;
- From a technical point of view, the XVA NMC GPU programming optimization techniques detailed in the appendix, including the innovative Algorithm 2 for efficient value-at-risk and expected shortfall computations;
- The proof of concept, by the numerical case studies of Sect. 4, that nested Monte Carlo XVA computations are within reach if GPU power is (properly) used (cf. Table 8).

2. XVA Guidelines

In this section we recall the XVA principles of Albanese and Crépey (2017), to which we refer the reader for more details. See also the companion papers by Albanese, Caenazzo, and Crépey (2017) and Armenti and Crépey (2017) for applications in respective bilateral and centrally cleared trading setups. For other XVA frameworks, see for instance Brigo and Pallavicini (2014) or Bichuch, Capponi, and Sturm (2017) (without KVA) or, with a KVA meant as an additional contra-asset like the CVA and the FVA (as opposed to a risk premium in our case), Green (2015), Green, Kenyon, and Dennis (2014), or Elouerkhaoui (2016).

We consider a pricing stochastic basis $(\Omega, \mathbb{F}, \mathbb{P})$, for a reference market filtration (ignoring the default of the bank itself) $\mathbb{F} = (\mathcal{F}_t)_{t\in\mathbb{R}_+}$ and a risk-neutral pricing measure $\mathbb{P}$ calibrated to vanilla market quotes, such that all the processes of interest are $\mathbb{F}$ adapted and all the random times of interest are $\mathbb{F}$ stopping times. This holds at least after so-called reduction of all the data to $\mathbb{F}$ starting from a larger filtration $\mathbb{G}$ including the default of the bank itself as a stopping time, supposing immersion from $\mathbb{F}$ into $\mathbb{G}$ for simplicity in this work.

Remark 2.1. Albanese and Crépey (2017) explicitly introduce the larger filtration $\mathbb{G}$ and show how to deal with the XVA equations, which are natively stated in $\mathbb{G}$, by reduction to $\mathbb{F}$ (denoting the reduced data with a $\cdot'$). Here we directly state the reduced equations in $\mathbb{F}$ (and we do not use the $\cdot'$ notation). ∎

The $\mathbb{P}$ expectation and $(\mathcal{F}_t, \mathbb{P})$ conditional expectation are denoted by $\mathbb{E}$ and $\mathbb{E}_t$. We denote by r an $\mathbb{F}$ progressive OIS (overnight indexed swap) rate process, which is together the best market proxy for a risk-free rate and the reference rate for the remuneration of the collateral. We write $\beta = e^{-\int_0^\cdot r_s ds}$ for the corresponding risk-neutral discount factor.

We consider a bank trading, in a bilateral and/or centrally cleared setup, with risky counterparties. The bank is also default prone, with default intensity γ and recovery rate R. We denote by T an upper bound on the maturity of all claims in the bank portfolio, also accounting for the time of liquidating the position between the bank and any of its clients in case of default.

By mark-to-market of a contract (or portfolio), we mean the (trade additive) risk-neutral conditional expectation of its future discounted promised cash flows D, ignoring counterparty risk and its capital and funding implications, i.e. without any XVAs. In particular, we denote by MtM the mark-to-market of the overall derivative client portfolio of the bank.

2.1. *Counterparty Exposure Cash Flows*

To mitigate counterparty risk, the bank and its counterparties post variation and initial margin as well as, in a centrally cleared setup, default fund contributions. The variation margin (VM) of each party tracks the mark-to-market of their portfolio at variation margin call times, as long as both parties are nondefault. However,

there is a liquidation period, usually a few days, between the default of a party and the liquidation of its portfolio. Even in the case of a perfectly variation-margined portfolio, the gap risk of slippage of the mark-to-market of the portfolio of a defaulted party and of unpaid contractual cash flows during its liquidation period justifies the need for initial margin (IM). The IM of each party is dynamically set as a risk measure, such as value-at-risk ($\mathbb{V}\mathrm{a}\mathbb{R}$) at some confidence level a_{im}, of their loss-and-profit at the time horizon of the liquidation period (sensitivity $\mathbb{V}\mathrm{a}\mathbb{R}$ in bilateral SIMM transactions and historical $\mathbb{V}\mathrm{a}\mathbb{R}$ for centrally cleared trading). In case a party defaults, its IM provides a buffer to absorb the losses that may arise on the portfolio from adverse scenarios, exacerbated by wrong-way risk, during the liquidation period.

On top of the variation and initial margins that are used in bilateral transactions, a central counterparty (CCP) deals with extreme and systemic risk on a mutualization basis, through an additional layer of protection, called the default fund (DF), which is pooled between the clearing members. The default fund is used when the losses exceed the sum between the VM and the IM of the defaulted member. The default fund contribution of the defaulted member is used first. If it does not suffice, the default fund contributions of the other clearing members are used in turn. Under the current European regulation, the Cover 2 EMIR rule requires to size the default fund as, at least, the maximum of the largest exposure and of the sum of the second and third largest exposures of the CCP to its clearing members, updated periodically (e.g. monthly) based on "extreme but plausible" scenarios. The corresponding amount is allocated between the clearing members, typically proportionally to their losses over IM (or to their IM itself).

2.2. *Funding Cash Flows*

Variation margin typically consists of cash that is re-hypothecable, meaning that received VM can be used for funding purposes and is remunerated OIS by the receiving party. Initial margin, as well as default fund contributions in a CCP setup, typically consist of liquid assets deposited in a segregated account, such as government bonds, which pay coupons or otherwise accrue in value. We assume that the bank can invest at the OIS rate r and obtain unsecured funding for borrowing VM at the rate $(r + \lambda)$, where the unsecured funding spread $\lambda = (1 - R)\gamma$ can be interpreted as an instantaneous CDS spread of the bank. Initial margin is funded separately from variation margin,[a] at a blended spread $\bar{\lambda}$ that depends on the IM funding policy of the bank.[b]

[a]See the third paragraph of Section 3.2 in Albanese et al. (2017)
[b]See Section 4.3 in Albanese et al. (2017).

2.3. *Cost of Capital Pricing Approach in Incomplete Counterparty Credit Risk Markets*

In theory, a bank may want to setup an XVA hedge. But, as (especially own) jump-to-default exposures are hard to hedge in pratice, such a hedge can only be very imperfect. In the context of XVA computations, we assume a perfectly collateralized back to back market hedge of its client portfolio by the bank (i.e. the bank posts MtM as variation margin on its hedge), but we conservatively assume no XVA hedge.

To deal with the corresponding market incompleteness issue, we follow a cost of capital XVA pricing approach, in two steps. First, the so-called contra-assets are valued as the expected costs of the counterparty default losses and risky funding expenses. Second, on top of these expected costs, a KVA risk premium (capital valuation adjustment) is computed as the cost of a sustainable remuneration of the shareholder capital at risk which is earmarked by the bank to absorb its exceptional (beyond expected) losses.

More precisely, the contra-asset value process (CA) corresponds to the expected discounted future counterparty default losses and funding expenditures. Incremental CA amounts are charged by the bank to its clients at every new deal and put in a reserve capital account, which is then depleted by counterparty default losses and funding expenditures as they occur.

In addition, bank shareholders require a remuneration at some hurdle rate h, commonly estimated by financial economists in a range varying from 6% to 13%, for the risk on their capital. Accordingly, an incremental risk margin (or KVA) is sourced from clients at every new trade in view of being gradually distributed to bank shareholders as remuneration for their capital at risk at rate h as time goes on.

Remark 2.2. In practice, target return on equities are fixed by the management of the bank every year. In the context of our XVA computations, we take h as an exogenous constant for simplicity.

Cost of capital calculations involve projections over decades in the future. The historical probability measure is hardly estimable on such time frames. As a consequence, we do all our price and risk computations under the risk-neutral measure $\mathbb{P}$.

The uncertainty on the hurdle rate or on the historical probability measure are left to model risk. ∎

2.4. *Contra-Assets Valuation*

We work under the modeling assumption that every bank account is continuously reset to its theoretical target value, any discrepancy between the two being instantaneously realized as loss or earning by the bank.

In particular, the reserve capital account of the bank is continuously reset to

its theoretical target CA level so that, much like with futures, the trading position of the bank is reset to zero at all times, but it generates a trading loss-and-profit process L. Our equation for the contra-assets value process CA is then derived from a risk-neutral martingale condition on the trading loss process L of the bank, along with a terminal condition $\mathrm{CA}_T = 0$. This martingale condition on L corresponds to a bank shareholder no arbitrage condition. It results in

$$\mathrm{CA} = \mathrm{CVA} + \mathrm{FVA} + \mathrm{MVA}, \tag{2.1}$$

for setup-dependent, but always nonnegative, CVA, FVA, and MVA processes. The FVA corresponds to the cost of funding cash collateral for variation margin, whereas the MVA is the cost of funding segregated collateral posted as initial margin.

We emphasize that we ignore the DVA and we only deal with nonnegative XVA numbers, in accordance with a shareholder-centric perspective where shareholders need be at least indifferent to a deal at a certain price for the deal to occur at that price. In case a DVA is needed (e.g. for regulatory accounting purposes), it can be computed much like the CVA. Regarding the funding issue, we consider an asymmetric, nonnegative FVA, which is part of an FVA/FDA accounting framework, as opposed to an FCA/FBA accounting framework where it is (unduly) assumed that the bank earns its credit spread when it invests excess cash generated from trading (see Albanese and Andersen (2014)).

Example 2.1. We consider a bank engaged into bilateral trading with a single client, with final maturity of the portfolio T. Let R_c denote the recovery rate of the client in case it defaults at time τ_c. Let PIM and RIM denote the initial margins posted and received by the bank on its client portfolio. Then, assuming for simplicity an instantaneous liquidation of the bank portfolio in case the client defaults, we have, for $0 \le t \le T$ (counting the VM positively when received by the bank):

$$\mathrm{CVA}_t = \mathbb{E}_t \mathbb{1}_{\{t<\tau_c\le T\}} \beta_t^{-1} \beta_{\tau_c} (1-R_c) \big(\mathrm{MtM}_{\tau_c} + D_{\tau_c} - D_{\tau_c -} - \mathrm{VM}_{\tau_c} - \mathrm{RIM}_{\tau_c}\big)^+, \tag{2.2}$$

$$\mathrm{FVA}_t = \mathbb{E}_t \int_t^{\tau_c \wedge T} \beta_t^{-1} \beta_s \lambda_s \Big(\mathrm{MtM}_s - \mathrm{VM}_s - \mathrm{CVA}_s - \mathrm{FVA}_s - \mathrm{MVA}_s\Big)^+ ds, \tag{2.3}$$

$$\mathrm{MVA}_t = \mathbb{E}_t \int_t^{\tau_c \wedge T} \beta_t^{-1} \beta_s \bar{\lambda}_s \mathrm{PIM}_s ds. \tag{2.4}$$

See Albanese et al. (2017) for the extension of these equations to the case of a bank engaged into bilateral trade portfolios with several counterparties. ∎

Note that the jump process $(D_{\tau_c} - D_{\tau_c -})$ of the contractually promised cash flows contributes to the CVA exposure of the bank

$$Q = \big(\mathrm{MtM} + D - D_- - \mathrm{VM} - \mathrm{RIM}\big) \tag{2.5}$$

that appears in (2.2), because $(D_{\tau_c} - D_{\tau_c -})$ fails to be paid by the client if it defaults. In most cases, however (with the notable exception of credit derivatives exposed in Sect. 4.3), this is immaterial because $D_{\tau_c} = D_{\tau_c -}$.

Remark 2.3. In the special case of a single counterparty, we also have the following equation for the CA = CVA + FVA + MVA process:

$$\begin{aligned}\mathrm{CA}_t = \mathbb{E}_t\Big[&\mathbb{1}_{\{t<\tau_c\le T\}}\beta_t^{-1}\beta_{\tau_c}(1-R_c)\big(\mathrm{MtM}_{\tau_c}+D_{\tau_c}-D_{\tau_c-}-\mathrm{VM}_{\tau_c}-\mathrm{RIM}_{\tau_c}\big)^+\\ &+\int_t^{\tau_c\wedge T}\beta_t^{-1}\beta_s\big(\lambda_s(\mathrm{MtM}_s-\mathrm{VM}_s-\mathrm{CA}_s)^+ + \bar\lambda_s\mathrm{PIM}_s\big)ds\Big],\ t\le t\le T.\end{aligned} \tag{2.6}$$

From the nested Monte Carlo perspective developed in Sect. 3, compared with (2.2), this equation "spares" the CVA and MVA Monte Carlo layer in the computation of the FVA. ∎

At the valuation time 0, assuming deterministic interest rates, the CVA (2.2) can be rewritten as

$$\mathrm{CVA}_0 = (1-R_c)\int_0^T \beta_t \mathrm{EPE}(t)\mathbb{P}(\tau_c\in dt), \tag{2.7}$$

where the expected positive exposure (EPE) is defined as

$$\mathrm{EPE}(t) = \mathbb{E}(Q_s^+|s=\tau_c)_{|\tau_c=t}.$$

As explained in Sect. 1, this identity is popular with practitioners as it decouples the credit and the market sides of the problem. But it is specific to the valuation time 0 and is only practical when the market and credit sides of the problem are independent, so that $\mathrm{EPE}(t) = \mathbb{E}(Q_t^+)$. It can hardly be extended rigorously to wrong-way risk (cf. Crépey and Song (2016)). A similar approach is commonly applied to FVA (and DVA) computations, with analogous pitfalls.

2.5. *Economic Capital and Capital Valuation Adjustment*

On top of no arbitrage in the sense of risk-neutral CA valuation, bank shareholders need be remunerated at some hurdle rate h for their capital at risk.

The economic capital (EC) of the bank is dynamically modeled as the conditional expected shortfall ($\mathbb{ES}$) at some quantile level a of the one-year-ahead loss of the bank, i.e., also accounting for discounting:

$$\mathrm{EC}_t = \mathbb{ES}_t^a\Big(\int_t^{t+1}\beta_t^{-1}\beta_s dL_s\Big). \tag{2.8}$$

As established in Albanese and Crépey (2017), Section 5.3, assuming a constant nurdle rate h, the amount needed by the bank to remunerate its shareholders for their capital at risk in the future is

$$\mathrm{KVA}_t = h\mathbb{E}_t\int_t^T e^{-\int_t^s (r_u+h)du}\mathrm{EC}_s ds,\ t\in[0,T]. \tag{2.9}$$

This formula yields the size of a risk margin account such that, if the bank gradually releases from this account to its shareholders an average amount

$$h(\mathrm{EC}_t - \mathrm{KVA}_t)dt, \tag{2.10}$$

then there is nothing left on the account at time T (if $T <$ the bank default time τ, whereas, if $\tau < T$, anything left on the risk margin account of the bank is instantaneously transferred to the creditors of the bank). The "$-\mathrm{KVA}_t$" in (2.10) or the "$+h$" in the discount factor in (2.9) reflect the fact that the risk margin is itself loss-absorbing and as such it is part of economic capital. Hence, shareholder capital at risk, which needs be remunerated at the hurdle rate h, only corresponds to the difference (EC − KVA). For simplicity we are skipping here the constraint that the economic capital must be greater than the ensuing KVA in order to ensure a nonnegative shareholder equity SCR=EC−KVA (cf. Albanese and Crépey (2017), Sections 5.4 and 7.2–7.3).

Remark 2.4. An alternative to economic capital in KVA computations is regulatory capital. Regulatory capital however is less consistent, as it loses the connection with other XVAs, whereby the input to EC and in turn KVA computations should be the loss and profit process L generated by the CVA, FVA, and MVA trading of the bank (a martingale part of the CA process in (2.2)). ∎

2.6. *Funds Transfer Price*

The total (or risk-adjusted) XVA is the sum of the risk-neutral CA and of the KVA risk premium. In the context of XVA computations, derivative portfolios are typically modeled on a run-off basis, i.e. assuming that no new trades will enter the portfolio in the future. Otherwise the bank could be led into snowball Ponzi schemes, whereby always more deals are entered for the sole purpose of funding previously entered ones. Moreover, the trade-flow of a bank, which is a price-maker, does not have a stationarity property that could allow the bank forecasting future trades.

Of course in reality a bank deals with incremental portfolios, where trades are added or removed as time goes on. In practice, incremental XVAs are computed at every new (or tentative) trade, as the differences between the portfolio XVAs with and without the new trade, the portfolio being assumed held on a run-off basis in both cases. The ensuing pricing, accounting, and dividend policy ensures the possibility for the bank to go into run-off, while staying in line with shareholder interest, from any point in time onward if wished.

3. Multi-layered NMC for XVA Computations

In this section we present the multi-layered dependence between the different XVA metrics and we discuss their nested Monte Carlo (NMC) implementation from a high-level perspective, referring the reader to the appendix for more technical GPU implementation and optimization developments.

3.1. *NMC XVA Simulation Tree*

The EC (2.8) and the IM are conditional risk measures. The KVA (2.9) and the MVA (2.4) are expectations of the latter integrated forward in time (or randomized, i.e. sampled at a random future time point). The FVA (2.3) is the solution of a backward SDE. The CVA (2.2) and MtM are conditional expectations.

In view of Remark 2.4 regarding the dependence of L with respect to the CVA, the MVA, and the FVA, a full Monte Carlo simulation of all XVAs would require a five layered NMC (six layers of Monte Carlo): The first and most inner layer would be dedicated to the simulation of MtM, then would come the IM, over which one would simulate the CVA and the MVA that are needed before simulating the FVA. The FVA layer must be run iteratively in time for solving the corresponding backward SDE on a coarse time grid. All previous quantities are used in the computation of the economic capital process involved in the most outer layer that approximates the KVA.

Figure 1 schematizes this interdependence between the simulation of MtM and of the different XVAs. The numbers of independent trajectories that have to be resimulated at each level are denoted by M_{kva}, M_{ec}, M_{fva}, M_{cva}, M_{im}, and M_{mtm}. For computational feasability, MtM and all XVAs should be functions of time t and of a Markovian risk factor vector process Z, which can generically be taken as a Markov chain H modulated by a jump-diffusion X, where the Markov chain component H encodes the default times τ_i of all the financial entities involved: cf. the model of Section 12.2.1 in Crépey (2013) (see also Section 4.2.1 there for a review of applications).

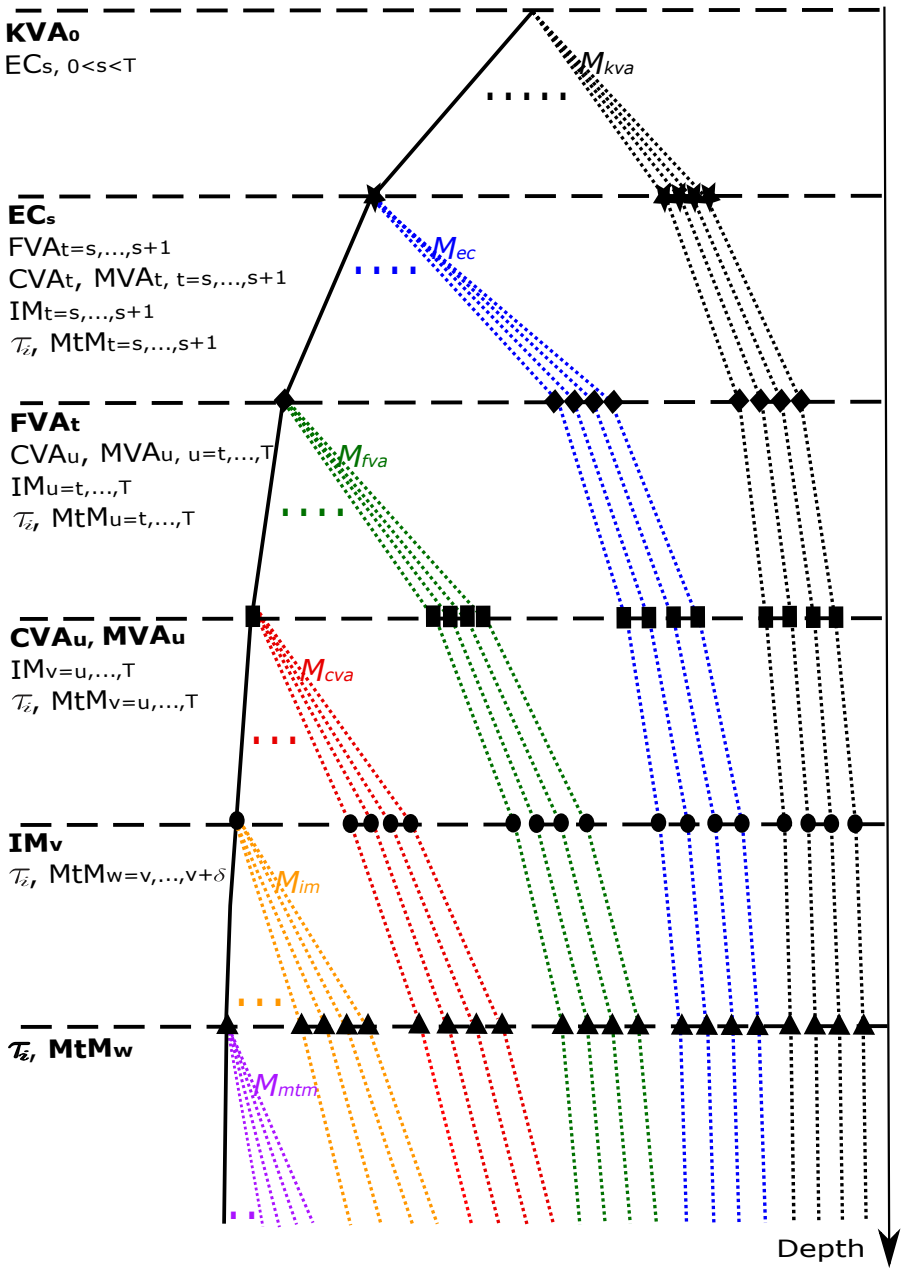

Fig. 1: XVA NMC simulation tree, from the most outer layer to the most inner one. The sub-tree rooted at the left-most node (on the solid path) on each line should be duplicated starting from each node on the right (on the dashed paths) on the same line.

Figure 1 yields the overall MtM and XVAs intra-dependency structure. However:

- If the user is only interested in some of the XVA components, then only the sub-tree corresponding to the highest XVA of interest in the figure needs be processed computationally;
- If one or several layers can be computed by explicit (exact or approximate) formulas instead of Monte Carlo simulation, then the corresponding layers drop out of the picture.

3.2. *Recursive Nested vs. Iterative Proxies*

The second bullet point above touches to the trade-off between iterative and recursive XVA computations, where some layers of the computation can either be recursively simulated in an NMC fashion, or iteratively pre-computed off-line and stored for further use in higher layers.

In an NMC perspective (see Gordy and Juneja (2010) for a seminal reference), higher layers are launched first and trigger nested simulations on-the-fly whenever required in order to compute an item from a lower layer. At the other extreme, in a purely iterative regression approach, the various layers are computed iteratively from bottom to top everywhere in time space—but with a non controllable error, as regression schemes only yield error control at the point where the regression paths are initiated. This is due to the unconditional approximations involved in regressions (unless advanced local regression basis techniques can be applied, cf. Gobet, Lemor, and Warin (2005)). Things get even worse with wrong-way risk extensions of such an approach or with its leveraging to higher order XVAs, when several regression layers are stacked one above the other, e.g. using a one-stage simulation with the same set of trajectories for regressing globally both MtM and some XVAs. The corresponding biases can then amplify each other, resulting in quite unpredictable numerical behavior.

Beyond the XVA area, such iterative vs. recursive trade-offs are an important NMC issue. Naive parametric approximations stacked one above the other as higher order XVAs are considered can only result in uncontrollable biases. Inner simulation in nested Monte Carlo instead creates variance, which is more manageable. Another alternative is the use of deterministic pricing schemes based on probability transition matrices as in Albanese and Li (2010) and Albanese et al. (2011). These can be very efficient and do not create any other than discretization bias, but they are restricted to certain models as discussed in Sect. 1.

In practice, as of today, playing with the two bullet points above, banks tend to put themselves in a position where at most one nested layer of Monte Carlo is required. For instance, 90% at least of their trading book consists of products with mark-to-market analytics, which spares the lowest layer in Figure 1. So far banks mostly rely on simple proxies for the KVA, such as an RWA or a regulatory KVA instead of a full-fledged economic capital based KVA (see Green, Kenyon, and Dennis (2014)). In the context of MVA simulations, they develop various tricks such as so-

called dynamic IM mapping in order to avoid resimulating for computing the IM at every future node (see Green and Kenyon (2015) or Anfuso, Aziz, Loukopoulos, and Giltinan (2017)). They avoid the BSDE feature of an asymmetric FVA by switching to a symmetric FVA corresponding to a conditional expectation for a suitably modified discount factor, which spares the iterated regressions in FVA computations (see Albanese et al. (2017, Remark 5.1), Albanese and Andersen (2014, 2015, 2015), and Andersen, Duffie, and Song (2017)). Above all, most banks are using an exposure-based XVA approach, described Sect. 1 and in the last paragraph of Sect. 2.4, where it is at most the determination of the mark-to-market cube that involves (simply layered) Monte Carlo simulation(/regression).

3.3. *XVA NMC Design Parameterization*

Based on sufficient regularity as well as a non-exploding moments condition expressed by their Assumption 1, Gordy and Juneja (2010) show that, for balancing the outer variance and square bias components of the mean square error in a one-layered NMC, the number of inner trajectories $M_{(1)}$ must be asymptotically of the order of the square root of the number of outer trajectories $M_{(0)}$. This is because, by a classical cancellation of the first order term in the related Taylor formula, the variance produced in the inner stage of a one-layered NMC is transformed into a bias in the outer stage. Hence, assuming that the variance of the outer random quantity is of the same order of magnitude as the conditional variances produced by the inner random quantities, an $M_{(0)} \otimes M_{(1)} = M_{(0)} \otimes \sqrt{M_{(0)}}$ NMC has the same $\mathrm{O}(M_{(0)}^{-\frac{1}{2}})$ accuracy as an $M_{(0)} \otimes M_{(1)} = M_{(0)} \otimes M_{(0)}$ NMC.

Leveraging this idea for a two-layered NMC with homogeneous variances, nested Taylor expansions yield that an $M_{(0)} \otimes M_{(0)} \otimes M_{(0)}$ NMC has the same order of accuracy as an $M_{(0)} \otimes M_{(0)} \otimes \sqrt{M_{(0)}}$ NMC, itself as accurate as an $M_{(0)} \otimes \sqrt{M_{(0)}} \otimes \sqrt{M_{(0)}}$ NMC. In the general case of an $i \geq 1$-layered NMC (e.g. $i = 5$ for the overall simulation of Figure 1) and assuming the same variance created through the different stages, $M_{(0)} \otimes M_{(1)} \otimes \ldots \otimes M_{(i)} = M_{(0)} \otimes \sqrt{M_{(0)}} \otimes \ldots \otimes \sqrt{M_{(0)}}$ has the same $\mathrm{O}(M_{(0)}^{-\frac{1}{2}})$ accuracy as $M_{(0)} \otimes M_{(0)} \otimes \ldots \otimes M_{(0)}$.

Moreover, in practice, the variance is not homogeneous with respect to the stages: For the $\mathbb{V}\mathrm{a}\mathbb{R}$ and $\mathbb{ES}$ of confidence level a that correspond to the IM and EC layers, the asymptotic rate of convergence is still given by the square root of the corresponding number of simulations (as in the case of an expectation), but this may come with larger constants (proportional to $(1-\alpha)^{-1}$ in particular, see Delmas and Jourdain (2006) and Embrechts, Klueppelberg, and Mikosch (1997)). For inner layers involving time iterated conditional averaging (space regressions), such as with inner FVA or Bermudan MtM computations, one may use only $\mathrm{O}(\sqrt{M_{(0)}}/\sqrt{N_b})$ or even $\mathrm{O}(\sqrt{M_{(0)}}/N_b)$ paths (depending on the regularity of the underlying cash flows, see Abbas-Turki and Mikou (2015)) without compromising the overall $\mathrm{O}(M_{(0)}^{-\frac{1}{2}})$ accuracy.

In view of the above, we propose an XVA NMC algorithm with the following design:

Input: The endowment of the bank, the credit curves of the bank and its clients, and, possibly, a new tentative client trade ;
Select layers of choice in a sub-tree of choice in Figure 1 (cf. Sect. 3.2), with corresponding tentative number of simulations denoted by $M_{(0)}, \ldots, M_{(i)}$, for some $1 \leq i \leq 5$ (we assume at least one level of nested simulation) ;
By dichotomy on $M_{(0)}$, reach a target relative error (in the sense of the outer confidence interval) for $M_{(0)} \otimes M_{(1)} \ldots \otimes M_{(i)}$ NMCs with $M_{(1)} = \ldots = M_{(i)} = \sqrt{M_{(0)}}$;
For each j decreasing from i to 1, reach by dichotomy on $M_{(j)}$ a target bias (in the sense of the impact on the outer confidence interval) for $M_{(0)} \otimes M_{(1)} \otimes \ldots \otimes M_{(j)} \otimes \ldots \otimes M_{(i)}$ NMCs ;
`Return(`*The time 0 XVAs of interest pertaining to the bank portfolio and, if relevant, the incremental XVAs pertaining to a tentative trade*`)`

Algorithm 1: XVA NMC algorithm.

Example 1. Considering the overall 5-layered NMC of Figure 1, in order to ensure a 5% relative error (in the sense of the corresponding confidence interval) at a 95% confidence level, which we can take as a benchmark order of accuracy for XVA computations in banks, the above approach may lead to M_{mtm}, M_{im}, and M_{cva} between $1e2$ and $1e3$. As the FVA is obtained from the resolution of a BSDE that involves multiple (conditional) averaging, M_{fva} can be even smaller than $1e2$ without compromising the accuracy. Due to the approximation of the conditional expected shortfall risk measure involved in economic capital computations, for M_{kva} of the order of $1e3$, M_{ec} has to be bigger than $1e3$ but usually can be smaller than $1e4$. ∎

3.4. *Coarse and Fine Parallelization Strategies*

Because of the wide exposure of a bank to a great number of counterparties, it would be very complicated and inefficient to divide the overall XVA computation into small pieces associated to local deals. In other words, it is unsuitable to distribute XVA computations: in the interest of data locality it is more appropriate to process the overall task in parallel. In fact, unless special cases (with a great deal of analytical tractability) are considered or crude and ad-hoc approximations without control error are used, the XVA NMC algorithm 1 can only be run on high performance concurrent computing architectures. Currently the most powerful ones involve GPUs (see Figure 3 in Section D.2). As of today, a state of the art GPU comprises roughly 4000 streaming processors operating at 2GHz each, versus 20 physical cores operating at 4GHz on a state of the art CPU. Hence a GPU implementation means a potential ~ 100 speedup with respect to a CPU implementation.

However, execution time is an output not only of the number and nature of computations (boiling down to bitsize opetrations, e.g. additions), but also of the number and nature of memory accesses (depending on the technology of the related physical storage capacities as well as on the distances that separate them from the computing units on the motherboard). In practice, the main bottleneck with GPU programming is memory and, more precisely, memory bandwidth (i.e. volume of data retrievable per second from the memory).

For XVA applications this advocates the use of supercomputers with fewer and fewer but bigger and bigger computing nodes[c] equipped with a very large memory and several GPUs. However, thanks to the NVLink technology that improves the communications between GPUs (cf. Nvidia (2017c)), scaling NMC from one GPU to various GPUs on a given node has become straightforward and very efficient. Moreover, NMC can be very well parallelized on different nodes with respect to outer trajectories, using the message passing interface (MPI) as explained in Abbas-Turki, Vialle, Lapreye, and Mercier (2009, 2014). Hence, nowadays, a single GPU implementation of NMC XVA computations is easily scalable to a supercomputer.

Regarding the parallelization strategy, from a GPU locality programming principle (see Nvidia (2017b, 2017a)):

- if only one GPU is available, then the paths should be allocated between its streaming processors from the most inner nested layer of simulation to the most outer one, in a fine grain stratification approach;
- if several GPUs on one single node are available, then one should allocate between the different GPUs the most outer paths of the simulation, using a fine grain stratification approach on each of them for the allocation of the computational task between its streaming processors;
- if even several computing nodes (each equipped with several GPUs) are available, then one should allocate between them the most outer paths of the simulation and, for each of them, allocate between the corresponding GPUs the intermediary levels of the simulation, using on each of them a fine grain stratification approach for allocating more inner computations between their respective streaming processors.

Remark 3.1. The first supporters of such a global procedure were Albanese et al. (2011), in the context, at that time, of CVA computations. With the advent of second generation XVAs in particular (MVA and KVA involving not only conditional expectations, but also conditional risk measures), data locality is even more stringent, but also much more accessible than before thanks to the large high bandwidth memories (HBM) available on GPUs and to the non-volatile memory architecture recently proposed by Intel. This favors the use of large memory computing nodes with a huge GPU computing throughput. As compared with Albanese et al. (2011)

[c]Basically, motherboards with 1 or 2 CPUs and 1 to 8 GPUs, see e.g. www.olcf.ornl.gov/summit/.

where backwardation pricing stages are treated by matrix exponentiation (in suitable models), the full NMC strategy of this paper is more complex, but it also more scalable to the above technology leaps.

4. Case Studies

In this section we illustrate the above by (single) nested Monte Carlo XVA computations in various markets and setups. The corresponding GPU programming optimization techniques are detailed in the Appendix.

All our simulations are run on a laptop that has an Intel i7-7700HQ CPU and a single GeForce GTX 1060 GPU programmed with the CUDA/C application programming interface (API).

4.1. *Common Shock Model of Default Times*

Under our approach in this paper, we favor a granular simulation of all defaults other than the default of the bank (which is treated by reduction of filtration as explained in Remark 2.1), as opposed to working with default intensities simply.

Regarding the default times τ_i, $i = 1, \dots, n$, of the financial entities (other than the bank) involved, we use the "common shock" or dynamic Marshall-Olkin copula (DMO) model of Crépey, Bielecki, and Brigo (2014), Chapt. 8–10 and Crépey and Song (2016) (see also Elouerkhaoui (2007, 2017). In this model defaults can happen simultaneously with positive probabilities.

First, we define shocks as pre-specified subsets of the credit names, i.e. the singletons $\{1\}, \{2\}, \dots, \{n\}$, for idiosyncratic defaults, and a small number of groups representing names susceptible to default simultaneously. For instance, a shock $\{1, 2, 4, 5\}$ represents the event that all the (non-defaulted entities among the) names 1, 2, 4, and 5 default at that time.

Given the family $\mathcal{Y}$ of shocks, the times τ_Y of the shocks $Y \in \mathcal{Y}$ are modeled as independent Cox times with intensity processes γ^Y (posibly dependent on the factor process X). For each credit name $i = 1, \dots, n$, we then set

$$\tau_i = \min_{\{Y \in \mathcal{Y};\, i \in Y\}} \tau_Y, \tag{4.1}$$

i.e. the default time of the name i is the first time of a shock Y that contains i. We write $J^i = \mathbb{1}_{[0,\tau_i)}$.

As demonstrated numerically Section 8.4 in Crépey et al. (2014), a few common shocks (beyond the idiosyncratic ones) are typically enough to ensure a good calibration of the model to market data regarding the credit risk of the financial entities and their default dependence (or to expert views about these).

The default model is then made dynamic, as required for XVA computations, by the introduction of the filtration of the indicator processes of the τ_i.

As detailed in Section A, we use a fine time discretization $\{t_0 = 0, \dots, t_{N_f} = T\}$ for the forwardation of the market risk factors and a coarse discretization $\{s_0 =$

$0, \ldots, s_{N_b} = T\}$ embedded in the latter for prices backwardation. In practice banks use adaptative time step, e.g. $\mathrm{h}_{k+1} = \mathrm{s}_{k+1} - \mathrm{s}_k$, reflecting the density of the underlying cash flows. In our case studies we just use uniform grids, fixing the number of time steps N_f used for a fine discretization of the factor process X (and then in turn of the Cox times used for generating the τ_Y) as a multiple of some coarse discretization parameter N_b. We simulate $M_{(0)}$ outer stage trajectories from $t_0 = 0$ to $t_{N_f} = T$. From each simulated value $\widehat{X}_k$, on the coarse time grid $k \in \{1, \ldots, N_b\}$, we simulate on the fine grid $M_{(1)}$ inner trajectories starting at $s_k = t_{kN_f/N_b}$ and ending at $t_{N_f} = T$, the final maturity of the portfolio. We denote by $(\widehat{Y}_k)$ any discretization on the coarse time grid (s_k) of a continuous time process (Y_t), by $(\widehat{Y}_k^j)$ independent simulations of the latter, and by $\widehat{\mathbb{E}}$ the empirical mean.

4.2. *CVA on Early Exercise Derivatives*

First we deal with CVA and the embedded MtM computations regarding a Bermudan option on a trivariate Black–Scholes process X.

We consider a Bermudan derivative, with mark-to-market $\mathrm{MtM}_t = P(t, X_t)$ for some pricing function $P = P(t, x)$ such that, at each pricing / exercise time s_k,

$$P(s_k, X_{s_k}) = \sup_{\theta \in \mathcal{T}^k} \mathbb{E}\left(\beta_{s_k}^{-1} \beta_\theta \phi(X_\theta) | X_{s_k}\right), \tag{4.2}$$

where ϕ is the payoff function and $\mathcal{T}^k$ is the set of stopping times valued in the coarse time subgrid $\{s_k, \ldots, s_{N_b}\}$.

For $k = 0, \ldots, N_b - 1$, let

$$A_k = \left\{\beta_{s_k}\phi(X_{s_k}) > \mathbb{E}\left(\beta_{s_{k+1}} P(s_{k+1}, X_{s_{k+1}}) \middle| X_{s_k}\right)\right\}.$$

As detailed in Clément, Lamberton, and Protter (2002), we have $P(0, x) = \mathbb{E}\left(\beta_{\tau_0^*}\phi(X_{\tau_0^*}) | X_0 = x\right)$, where τ_0^* is defined through the iteration $\tau_{N_b}^* = T$ and, for k decreasing from $(N_b - 1)$ to 0,

$$\tau_k^* = s_k \mathbb{1}_{A_k} + \tau_{k+1}^* \mathbb{1}_{A_k^c}. \tag{4.3}$$

The computation of $\mathrm{MtM}_0 = P(0, X_0 = x)$ can then be performed by the simulation/regression algorithm of Longstaff and Schwartz (2001), which consists in approximating the conditional expectation involved in A_k by regression on a basis of monomial functions. Moreover, in an NMC setup, this can be extended to the estimation of any $\mathrm{MtM}_{s_{k+1}} = P(s_{k+1}, X_{s_{k+1}})$ in the obvious way, by nested simulation rooted at $(s_{k+1}, X_{s_{k+1}} = x)$.

Let $\tau_1 = \tau_c$ represent the default time of a client of the bank with default intensity equal to $\gamma^c = 100$bp and recovery $R_c = 0$ (here $n = 1$, i.e. there is only one credit name involved). The process X is a three dimensional Black & Scholes model with common volatilities $\sigma_{i=1,2,3} = 0.2$ and pairwise correlations $= 50\%$. The spot value $X_0^{i=1,2,3} = 100 = K$, where K is the strike of a Bermudan put on the average with maturity $T = 1$. We assume no (variation or initial) margins.

By (2.2), the time 0 CVA of the option is approximated as

$$\mathrm{CVA}_0 \approx \sum_{k=0}^{N_b-1} \mathbb{E}\Big(\mathbb{1}_{\{\tau_c\in(s_k,s_{k+1}]\}}\beta_{s_{k+1}}(1-R_c)\mathrm{MtM}^+_{s_{k+1}}\Big),$$

which we estimate by

$$\widehat{\mathrm{CVA}}_0 = \frac{1}{M_{cva}}\sum_{j=1}^{M_{cva}}\sum_{k=0}^{N_b-1}\mathbb{1}_{\{\tau_c^j\in(s_k,s_{k+1}]\}}\widehat{\beta}^j_{k+1}(\widehat{\mathrm{MtM}}^j_{k+1})^+, \tag{4.4}$$

where the $\widehat{\mathrm{MtM}}^j_{k+1}$ are computed by nested Longstaff-Schwartz algorithms as explained above.

Using the GPU optimizations presented in Sections A, B.2, and D.1 , within less than a tenth of a second, this procedure yields the results presented in Table 1 for the computation of $\widehat{\mathrm{CVA}}_0$. According to Table 1, for $M_{cva} = 4096$, it is not necessary to simulate more than $M_{mtm} = 128$ inner trajectories if one targets an error of ± 5, as the corresponding estimates for $M_{mtm} = 128$ and 256 only differ by 1.

M_{mtm}	CVA_0 value
32	348
128	334
512	333

Table 1: CVA_0 for an Bermudan put of payoff $(K-\frac{1}{3}(x_1+x_2+x_3))^+$: $M_{cva} = 4096$ and confidence interval 95% of ± 5.

4.3. *CVA and FVA on Defaultable Claims*

Next we deal with CVA, FVA, and the embedded MtM computations relative to a portfolio of credit default swaps (CDS).

In this case, $\tau_1 = \tau_c$ represents the default time of a single financial counterparty (client) of the bank, with recovery rate R_c, and $\tau_2, \ldots, \tau_n$ are the default times of reference financial entities in credit default swaps between the bank and its client.

We consider stylized CDS contracts corresponding to cash flow processes of the form, for $i \geq 2$ and $0 \leq t \leq T$:

$$D^i_t = Nom_i \times \big((1-R_i)\mathbb{1}_{t\geq\tau_i} - S_i(t\wedge\tau_i)\big), \tag{4.5}$$

where all recoveries R_i are set to 40% and all nominals Nom_i are set to 100.

A portfolio of 70 payer CDS contracts and 30 receiver CDS contracts, as well as all the numerical parameters for the simulations, are taken as in Crépey et al. (2014), Section 10.1.2, in the case of a client with CDS spread equal to 100bp. In particular, no variation or initial margins are assumed.

We denote by $\mathrm{MtM}^i = \mathbb{E}_t[\int_t^T \beta_t^{-1}\beta_s dD_s^i]$ the mark-to-market of a fees-payer CDS on name i, so that the mark-to-market of the overall portfolio is

$$\mathrm{MtM} = \left(\sum_{i\ pay} - \sum_{i\ rec}\right)\mathrm{MtM}^i. \tag{4.6}$$

We also consider the pre-default pricing functions $P^i(t,x)$ such that $\mathrm{MtM}_t^i = J_t^i P^i(t, X_t)$ (cf. Proposition 13.4.1 in Crépey, Bielecki, and Brigo (2014)).

The contractual spreads S_i are such that the CDS contracts break even at time 0, i.e $\mathrm{MtM}_0^i = 0$. Setting $\mathrm{LGD}_i = (1-R_i)Nom_i$, we denote

$$\Delta_t = \Big(\sum_{i\ \mathrm{pay}} - \sum_{i\ \mathrm{rec}}\Big)(D_t^i - D_{t-}^i) = \Big(\sum_{i\ \mathrm{pay}} - \sum_{i\ \mathrm{rec}}\Big)\mathbb{1}_{\tau_i=t<T}\mathrm{LGD}_i,$$

the jump process of the cash flows priced by MtM (as explained after (2.5), the process Δ belongs to the CVA exposure).

In order to realistically mimic the randomness of the CDS credit spreads, we use the affine intensity framework of Crépey et al. (2014), Section 10.1.1, for the intensities γ^Y in the common shocks model, based on a multivariate CIR (component-wise) factor process $X = (X^1, X^2, X^3)$.

4.3.1. *CVA*

We have the following representation for the time 0 CVA (cf. (2.2)):

$$\begin{aligned}\mathrm{CVA}_0 \approx \sum_{k=0}^{N_b-1} \mathbb{E}\Big(&\mathbb{1}_{\{\tau_c\in(s_k,s_{k+1}]\}}\beta_{s_{k+1}}(1-R_c)\times \\ &\big[\Big(\sum_{i\ pay} - \sum_{i\ rec}\Big)\mathbb{1}_{\{\tau_i>s_{k+1}\}}P^i(s_{k+1}, X_{s_{k+1}}) + \Big(\sum_{i\ pay} - \sum_{i\ rec}\Big)\mathbb{1}_{\{\tau_i\in(s_k,s_{k+1}]\}}\mathrm{LGD}_i\big]^+\Big),\end{aligned}$$

Denoting by $(\widehat{P}_k^{i,j}, \widehat{\tau}_i^j)$ simulated values of the $P^i(s_k, X_{s_k})$, estimated using inner trajectories, and of the τ_i, our estimator $\widehat{\mathrm{CVA}}_0$ of (4.3.1) is given by (recall $\tau_1 = \tau_c$)

$$\begin{aligned}\widehat{\mathrm{CVA}}_0 \quad &= \frac{1}{M_{cva}}\sum_{j=1}^{M_{cva}}\sum_{k=0}^{N_b-1}\mathbb{1}_{\{\widehat{\tau}_1^j\in(s_k,s_{k+1}]\}}\widehat{\beta}_{k+1}(1-R_c)\times \\ &\qquad\left[\Big(\sum_{i\ pay} - \sum_{i\ rec}\Big)\mathbb{1}_{\{\widehat{\tau}_i^j>s_{k+1}\}}\widehat{P}_{k+1}^{i,j} + \Big(\sum_{i\ pay} - \sum_{i\ rec}\Big)\mathbb{1}_{\{\widehat{\tau}_i^j\in(s_k,s_{k+1}]\}}\mathrm{LGD}_i\right]^+.\end{aligned}$$

Table 2 shows the results of a GPU implementation of the above benefiting from the optimizations presented in Sections A and B.2. We see that, for $M_{cva} = 1024 * 100$ paths, taking $M_{mtm} = 128$ is already enough: The gain in bias that results from taking M_{mtm} greater is negligible with respect to the uncertainty of the simulation (size of the confidence interval).

M_{mtm}	CVA value	CI 95%	Rel. err.	Exec. time (sec)
128	2358.92	±76.47	3.24%	5.96
256	2358.46	±76.45	3.24%	11.54
512	2367.90	±76.49	3.23%	23.18
1024	2373.83	±76.51	3.22%	46.12

Table 2: CVA: $M_{cva} = 1024 * 100$, counterparty spread = 100bp.

4.3.2. *CA BSDE*

We now move to FVA computations.

Assuming no margins on client trades and an instantaneous liquidation of defaulted names, the BSDE (2.6) for the CA = CVA + FVA process reads as

$$\begin{aligned}\beta_t \mathrm{CA}_t = \mathbb{E}_t\Big[&\beta_{\tau_c}\mathbb{1}_{\{t<\tau_c<T\}}(1-R_c)(\mathrm{MtM}_{\tau_c}+\Delta_{\tau_c})^+ \\ &+\int_t^{\tau_c\wedge T}\beta_s\lambda(\mathrm{MtM}_s-\mathrm{CA}_s)^+ds\Big],\ t\in[0,T].\end{aligned} \tag{4.7}$$

For $t \in [0,T]$, let

$$cva_t = (1-R_c)\sum_{Y\in\mathcal{Y};1\in Y}\gamma_t^Y\Big(\big(\sum_{i\ \mathrm{pay},i\notin Y}-\sum_{i\ \mathrm{rec},i\notin Y}\big)(\mathrm{MtM}_t^i+\mathbb{1}_{\tau_i=t<T}\mathrm{LGD}_i)\Big)^+$$

$$f_t(y) = \mathrm{cva}_t + \lambda(\mathrm{MtM}_t - y)^+ - (r_t+\gamma_t^c)y.$$

Remark 4.1. Estimating $f_t(y)$ requires a Monte Carlo simulation for evaluating the embedded MtM_t^i. Hence, in an outer simulation context, obtaining an estimate $\widehat{f_k}(y)$ of $f_{s_k}(y)$ requires nested simulation. ∎

Following the approach detailed in Crépey and Song (2016), the original CA BSDE (4.7) on $[0,\tau_c\wedge T]$ is equivalent to the reduced BSDE on $[0,T]$ given by

$$\mathrm{CA}_t = \mathbb{E}_t\int_t^T f_s(\mathrm{CA}_s)ds, t\in[0,T]\,, \tag{4.8}$$

i.e. the solutions to (4.7) and (4.8) (both well-posed under mild technical conditions) coincide before the client default time $\tau_c = \tau_1$ and, in particular, at the valuation time 0.

The reduced CA BSDE (4.8) is approximated on the coarse time grid, with time steps denoted by h_{k+1} (a constant in our numerics), by $\mathrm{CA}_T = 0$ and, for k decreasing from (N_b-1) to 0,

$$\mathrm{CA}_{s_k} \approx \mathbb{E}_{s_k}\left(\mathrm{CA}_{s_{k+1}} + \mathrm{h}_{k+1}f_{s_{k+1}}(\mathrm{CA}_{s_{k+1}})\right), \tag{4.9}$$

where the conditional expectation is estimated by regression.

Remark 4.2. For dealing with FVA semilinearity, alternatives to time iterated regressions in space are the marked branching diffusion approach of Henry-Labordère

(2012) or[d] the expansion technique of Fujii and Takahashi (2012a, 2012b): see Crépey and Nguyen (2016) for a comparison between these approaches in a single layer, non nested Monte Carlo setup.

As our forward factor process Z is very high-dimensional (three CIR processes X plus the default indicator process of each CDS reference credit name), by principal component analysis (PCA), we filter out the few eigenvectors associated with the numerically null eigenvalues, to make the PCA numerically well-posed, and we reduce further the dimension of the regression to accelerate it. In practice, keeping 90% to 95% of the information spectrum (total variance, i.e. sum of the eigenvalues) is found to require no more than half of the eigenvalues, and even much less when the time step is getting away from the maturity (cf. the right panel in Figure 8). We denote by $\widehat{\Psi}_k$, $\widehat{\Lambda}_k$, and $\widehat{\Gamma}_k$ the reduced regression basis, eigenvalues vector, and eigenvectors matrix that are used at each (coarse) regression time s_k. Let

$$\Phi_{s_{k+1}} = \mathrm{CA}_{s_{k+1}} + \mathrm{h}_{k+1} f_{s_{k+1}}(\mathrm{CA}_{s_{k+1}}) - \mathbb{E}\big(\mathrm{CA}_{s_{k+1}} + \mathrm{h}_{k+1} f_{s_{k+1}}(\mathrm{CA}_{s_{k+1}})\big),$$
$$\widehat{\Phi}_{k+1} = \widehat{\mathrm{CA}}_{k+1} + \mathrm{h}_{k+1} \widehat{f}_{k+1}(\widehat{\mathrm{CA}}_{k+1}) - \widehat{\mathbb{E}}\big(\widehat{\mathrm{CA}}_{k+1} + \mathrm{h}_{k+1} \widehat{f}_{k+1}(\widehat{\mathrm{CA}}_{k+1})\big).$$

A centered approximation to (4.9) is given by

$$\mathbb{E}\big(\Phi_{s_{k+1}} | Z_{s_k}\big) \simeq \ \widehat{B}_k^{\mathsf{T}} \widehat{\Psi}_k,$$

where $\cdot^{\mathsf{T}}$ denotes matrix transposition and

$$\widehat{B}_k = \widehat{\Lambda}_k^{-1} \widehat{\Gamma}_k^{\mathsf{T}} \widehat{\mathbb{E}}(\widehat{\Phi}_{k+1} \widehat{Z}_k), \ \widehat{\Psi}_k = \ \widehat{\Gamma}_k^{\mathsf{T}} \widehat{Z}_k.$$

The ensuing CA estimator is defined by $\widehat{\mathrm{CA}}_{N_b} = 0$ and, for $k = N_b - 1, \ldots, 2$,

$$\widehat{\mathrm{CA}}_k = \frac{1}{M_{fva}} \sum_{j=1}^{M_{fva}} [\widehat{\mathrm{CA}}_{k+1}^j + \mathrm{h}_{k+1} \widehat{f}_{k+1}^j(\widehat{\mathrm{CA}}_{k+1}^j)] + \ \widehat{B}_k^{\mathsf{T}} \widehat{\Psi}_k,$$

followed by

$$\widehat{\mathrm{CA}}_0 = \frac{1}{M_{fva}} \sum_{j=1}^{M_{fva}} \big(\widehat{\mathrm{CA}}_1^j + \mathrm{h}_1 \widehat{f}_1^j(\widehat{\mathrm{CA}}_1^j)\big),$$

where each $\widehat{f}_{k+1}^j(\widehat{\mathrm{CA}}_{k+1}^j)$ requires a nested simulation for evaluating the embedded $\mathrm{MtM}_{s_{k+1}}$ (cf. Remark 4.1).

Tables 3 and 4 show the results of a GPU implementation of the above using the optimizations of Sections A and D.2.

M_{mtm}	FVA value	CI 95%	Rel. err.	Exec. time (sec)
128	1861.66	±52.03	2.79%	81.21
256	1861.96	±52.03	2.79%	162.12
512	1861.83	±52.03	2.79%	324.67

[d]Modulo the regularity issue related to Lipchitz only coefficients as discussed in Gobet and Pagliarani (2015).

Table 3: FVA: $M_{fva} = 128 * 100$, $\lambda = 50$bp, keeping 90% of the information in the regressions for the conditional expectations.

M_{mtm}	FVA value	CI 95%	Rel. err.	Exec. time (sec)
128	1886.60	±53.75	2.87%	81.41
256	1886.73	±53.75	2.87%	162.52
512	1886.91	±53.75	2.87%	324.97

Table 4: FVA: $M_{fva} = 128 * 100$, $\lambda = 50$bp, keeping 95% of the information in the regressions for the conditional expectations.

The third and fourth columns in Table 3 show that, when 90% of the spectrum is kept in the regressions for the conditional expectations, $M_{fva} = 128 * 100$ outer simulations are enough to ensure a reasonable accuracy. The second column shows that the FVA values are then already stabilized after a low number $M_{mtm} = 128$ of inner simulations.

Table 4 shows the analogous results when 95% of the information is kept in the regressions. As the confidence interval in the previous 90% case is far greater than the impact on the FVA estimate of switching from 90% to 95%, we conclude that keeping 90% of the spectrum is sufficient.

4.4. *KVA on Swaps*

Finally we consider the time 0 KVA and the embedded EC computations in the context of the sizing and allocation of the default fund of a CCP (see the last paragraph in Sect. 2.1). All the lower layers in Figure 1 are analytical in the considered (Black-Scholes) model, so that the computation is simply nested. The GPU implementation involves the optimizations presented in Sections A, B.1, and C.

We consider a CCP with $(n+1)$ clearing members, labeled by $i = 0, 1, 2, \ldots, n$. We denote by:

- T: an upper bound on the maturity of all claims in the CCP portfolio, also accounting for a constant time $\delta > 0$ of liquidating the position between the bank and any of its counterparty in case of default;
- D_t^i: the cumulative contractual cash flow process of the CCP portfolio of the member i, cash flows being counted positively when they flow from the clearing member to the CCP;
- $\mathrm{MtM}_t^i = \mathbb{E}_t[\int_t^T \beta_t^{-1}\beta_s dD_s^i]$: the mark-to-market of the CCP portfolio of the member i;
- τ_i and $\tau_i^\delta = \tau_i + \delta$: the default and liquidation times of the member i, with nondefault indicator process $J^i = \mathbb{1}_{[0,\tau_i)}$;
- $\Delta^i_{\tau_i^\delta} = \int_{[\tau_i,\tau_i^\delta]} \beta_t^{-1}\beta_s dD_s^i$: the cumulative contractual cash flows of the member i, accrued at the OIS rate, over the liquidation period of the clearing member i;

- $\mathrm{VM}^i_t, \mathrm{IM}^i_t \geq 0$: VM and IM posted by the member i at time t.

By contrast with the Cover 2 rule, which is purely market risk based, Armenti and Crépey (2017) study a broader risk-based specification for the sizing of the default fund, in the form of a risk measure of the one-year ahead loss-and-profit of the CCP if there was no default fund—loss-and-profit as it results from the combination of the credit risk of the clearing members and of the market risk of their portfolios. Such a specification can be used for allocating the default fund between the clearing members, after calibration of the quantile level a_{df} below to the Cover 2 regulatory size at time 0.

Specifically, we define the loss process of a CCP that would be in charge of dealing with member counterparty default losses through a CVA^{ccp} account (earning OIS) and capital at risk at the aggregated CCP level as, for $t \in (0,T]$ (starting from some arbitrary initial value, since it is only the fluctuations of L^{ccp} that matter, and denoting by $\boldsymbol{\delta}_{\tau_i^\delta}$ a Dirac measure at time τ_i^δ),

$$\begin{aligned} \beta_t dL^{ccp}_t = &\sum_i \left(\beta_{\tau_i^\delta}(\mathrm{MtM}^i_{\tau_i^\delta} + \Delta^i_{\tau_i^\delta}) - \beta_{\tau_i}(\mathrm{VM}^i_{\tau_i} + \mathrm{IM}^i_{\tau_i})\right)^+ \boldsymbol{\delta}_{\tau_i^\delta}(dt) \\ &+ \beta_t(d\mathrm{CVA}^{ccp}_t - r_t \mathrm{CVA}^{ccp}_t)dt \end{aligned} \tag{4.10}$$

(and L^{ccp} constant from time T onward), where the CVA of the CCP is given as

$$\mathrm{CVA}^{ccp}_t = \mathbb{E}_t \sum_{t<\tau_i^\delta<T} \beta_t^{-1}\left(\beta_{\tau_i^\delta}(\mathrm{MtM}^i_{\tau_i^\delta} + \Delta^i_{\tau_i^\delta}) - \beta_{\tau_i}(\mathrm{VM}^i_{\tau_i} + \mathrm{IM}^i_{\tau_i})\right)^+, \ 0 \leq t \leq T.$$

The ensuing economic capital process of the CCP is (cf. (2.8))

$$\mathrm{EC}^{ccp}_t = \mathbb{ES}^{a_{df}}_t \left(\int_t^{t+1} \beta_t^{-1} \beta_s dL^{ccp}_s \right), \tag{4.11}$$

which yields the size of an overall risk based default fund at the confidence quantile level a_{df}. Note that, in view of (4.10), we have in (4.11):

$$\begin{aligned} \int_t^{t+1} \beta_s dL^{ccp}_s = &\sum_{t<\tau_i^\delta\leq t+1} \left(\beta_{\tau_i^\delta}(\mathrm{MtM}^i_{\tau_i^\delta} + \Delta^i_{\tau_i^\delta}) - \beta_{\tau_i}(\mathrm{VM}^i_{\tau_i} + \mathrm{IM}^i_{\tau_i})\right)^+ \\ &- (\beta_t \mathrm{CVA}^{ccp}_t - \beta_{t+1}\mathrm{CVA}^{ccp}_{t+1}). \end{aligned} \tag{4.12}$$

The KVA of the CCP estimates how much it would cost the CCP to remunerate all clearing members at some hurdle rate h for their capital at risk in the default fund, namely, for $t \leq T$ (cf. (2.9)),

$$\mathrm{KVA}^{ccp}_t = h\mathbb{E}_t \left[\int_t^T \beta_s e^{-hs} \mathrm{EC}^{ccp}_s ds \right], \tag{4.13}$$

which we estimate at time 0 by

$$\widehat{\mathrm{KVA}}^{ccp}_0 = \frac{h}{M_{kva}} \sum_{j=1}^{M_{kva}} \sum_{k=0}^{N_b-1} \widehat{\beta}_{k+1} e^{-hs_{k+1}} \widehat{\mathrm{EC}}^{ccp}_{k+1} \mathrm{h}_{k+1}. \tag{4.14}$$

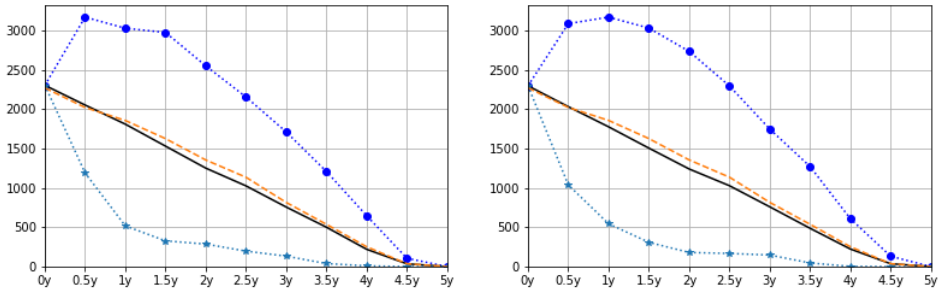

Fig. 2: Economic capital processes: '- -' unconditional, '· · · ∗' quantile 1%, '—' mean, '· · · ●' quantile 99%. Left: $M_{kva} = 256$ and $M_{ec} = 32 * 100$. Right: $M_{kva} = 1024$ and $M_{ec} = 32 * 100$.

For our simulations we consider the CCP toy model of Armenti and Crépey (2017), Section 4, where $n + 1 = 9$ members are clearing foreign exchange swaps on a Black underlying, with all the numerical parameters used there (in particular, $a_{df} = 99\%$). The time 0 KVA^{ccp} in (4.14) is computed in two alternative ways:

- By a GPU nested Monte Carlo procedure using the optimized sorting procedure of Section C for the inner $\widehat{\text{EC}}^{ccp}_{k+1}$ computations;
- By a non-nested proxy approach where $\mathbb{ES}^{a_{df}}_t$ is replaced by $\mathbb{ES}^{a_{df}}_0$ in (4.11) and $\widehat{\text{EC}}^{ccp}_{k+1}$ is replaced by the corresponding (deterministic) term structure, which we denote by $\widetilde{\text{EC}}^{ccp}_{k+1}$, in (4.14).

We refer the reader to Equation (33) and the ensuing discussion in Albanese et al. (2017) for the detail of the second procedure, motivated by the fact that, for the computation of the KVA, an outer (unconditional) expectation will be applied anyway: The comparison between Tables 5 and 6–7 shows that the ensuing bias on the resulting time 0 KVA estimate, denoted by $\widetilde{\text{KVA}}^{ccp}_0$, may be acceptable, bearing in mind the model risk intrinsic to such quantities.

Regarding the nested computation, Table 6 shows that the variance is acceptable already for a low number, such as $M_{kva} = 512$, of outer simulations. Table 7 shows that this is not specific to the relatively high number $M_{ec} = 1024 \times 100$ of inner simulations used in Table 6: as we can see from Table 7, for $M_{kva} = 1024$, the KVA value is already stabilized for $M_{ec} = 128 \times 100$.

M_{kva}	$\widetilde{\mathrm{KVA}_0}^{ccp}$
$128 * 100$	518.44
$256 * 100$	515.49
$512 * 100$	512.94
$1024 * 100$	508.96

Table 5: KVA_0 estimated by deterministic projections of economic capital, without nested simulation.

M_{kva}	$\widehat{\mathrm{KVA}_0}^{ccp}$	CI 95%	Rel. err.	Exec. time (sec)
256	493.68	± 11.80	2.39%	0.09
512	487.53	± 8.82	1.80%	0.17
1024	489.03	± 6.07	1.24%	0.34
2048	490.33	± 4.19	0.85%	0.73
4096	491.55	± 3.01	0.61%	1.24

Table 6: KVA_0 estimated by nested simulation: $M_{ec} = 32 * 100$.

M_{ec}	$\widehat{\mathrm{KVA}_0}^{ccp}$ value	Exec. time (sec)
$8 * 100$	479.89	0.09
$16 * 100$	485.81	0.21
$32 * 100$	489.03	0.46
$64 * 100$	490.39	1.09
$128 * 100$	491.32	3.14

Table 7: KVA_0 estimated by nested simulation: $M_{kva} = 1024$.

4.5. *Speedup Table*

GPU optimizations, regarding a proper use of the different available memory layers in particular, are important for effectively benefiting from the target ~ 100 speedup factor when compared with an optimized CPU implementation. Table 8 summarizes the speedups obtained in our case studies thanks to the various GPU optimizations detailed in the appendix. The rows of the table are labeled as the corresponding sections in the appendix.

First, the computation times also displayed in the table show that NMC computations are within reach provided GPU and the related optimizations are used, even for computations involving portfolios of one hundred credit names, time backwardation of nonlinearities, or conditional risk measure computations: Although the corresponding two-stage Monte Carlos would be quite heavy to implement on a CPU, the GPU implementations, suitably optimized, are quite fast. Additional speedups would readily follow from the use of several GPUs as explained in Sect. 3.4.

	CVA		FVA		KVA	
	Time	Speedup	Time	Speedup	Time	Speedup
A. Nested simulation	5.80	3.5	80.51	3.5	0.96	3.5
B.1. Sorting defaults					0.22	1.2
B.2. Listing defaults	0.06	1.4				
C. $\mathbb{V}$a$\mathbb{R}$ and $\mathbb{ES}$					0.12	20
D.1. Inner regressions	0.06	2				
D.2. Outer regressions			0.05	3		
Other	0.1		0.64		0.11	

Table 8: Execution times (in seconds) and GPU optimization speedups in the NMC case studies of Sect. 4.2 through 4.4: CVA for $M_{cva} = 1024 * 100$ and $M_{mtm} = 128$; FVA for $M_{fva} = 128 * 100$ and $M_{mtm} = 128$; KVA for $M_{kva} = 128$ and $M_{ec} = 1024 * 100$.

The second take-away message, which corresponds to an "horizontal reading" of Table 8, is that the optimization speedups are quite significant, as expected. We emphasize that we are only considering here speedups internal to a GPU implementation more or less optimized in a way or another as detailed in the appendix, we do not report on any CPU implementation execution times (that would be much slower) or GPU versus CPU speedups (as an optimized GPU implemetation obviously dominates an optimized CPU implementation).

Last, one should not overdue a "vertical reading" of the table to the conclusion, for instance, that defaults' listing and sorting would be less important than the other optimizations or that the inner regression phase is much faster than the nested simulation of paths. This is model dependent and restricted to the setup and implementation of the case studies that underlie all numbers in Table 8. Thus, a careful reading of Section B shows that the relative importance of the defaults' listing and sorting optimizations strongly depends on the default structure of the considered model. Likewise, a numerically more robust implementation of the inner regressions on few paths (hence prone to ill-conditioning, numerical instabilities and sensitivity to roundoff errors), based on a spectral decomposition of the covariance matrices that are repeatedly generated along inner simulation/regression stages (see the end of Section D.1 and cf. also Sect. 4.3.2), could easily be 10 to 20 times slower than Cholesky that was used in the inner regression case study of Sect. 4.2. Etc..

5. Perspectives

Due to the number of market input data involved in the calibration of a real-life XVA engine, the consideration not only of the XVAs themselves, but also of all the corresponding bump sensitivities, may easily increase the XVA computational burden by a factor of ~ 1000. Hence, another interesting topic left aside in this paper for length sake is XVA Greeks and their GPU acceleration, whether performed by

the maximum likelihood method (see Albanese and Li (2010)), by (hard to handle memory-wise though) adjoint differentiation (see Reghai, Kettani, and Messaoud (2015)), or through various regression techniques. As initial margins, at least in a bilateral SIMM setup, involve a sensitivity $\mathbb{V}\mathrm{a}\mathbb{R}$, this is also the reason why we did not show any MVA computations in our case studies (MVA in a centrally cleared setup however can be tackled by similar tokens as the KVA of Sect. 4.4).

By their nature, XVA computations appeal very naturally to nested Monte Carlo computations. However, whenever non-(or less-)nested Monte Carlo schemes are available without bias, they should be preferred to a nested alternative. In particular, the parameterized stochastic approximation or "quasi-regression" algorithm[e] of Crépey, Fort, Gobet, and Stazhynski (2017), initially conceived for dealing with model uncertainty, could also be used for reconstructing a conditional risk measure such as $\mathbb{V}\mathrm{a}\mathbb{R}$ (for IM computations) or $\mathbb{ES}$ (for EC computations) as a function of the risk factors (provided their number can be kept reasonable), with global error control (cf. Sect. 3.1). To reduce further the complexity of the NMC structure, one might also think of multi-level techniques à la Giles (2008).

Banks can use their economic capital as a funding source for variation margin. Accounting for this feature results in the following modified FVA formula, instead of (2.3) :

$$\mathrm{FVA}_t = \mathbb{E}_t \int_t^T \beta_t^{-1}\beta_s\lambda_s\Big(\mathrm{MtM}_s - \mathrm{VM}_s - \mathrm{CVA}_s - \mathrm{FVA}_s - \mathrm{MVA}_s - \mathrm{EC}_s(L)\Big)^+ ds,$$

where we recall from Sect. 2.4 that the trading loss process L is the martingale part of the $\mathrm{CA} = \mathrm{CVA} + \mathrm{FVA} + \mathrm{MVA}$ process. The ensuing XVA equations "of the McKean type" are shown to be well-posed in Crépey, Élie, Sabbagh, and Song (2017). To solve them numerically, Albanese et al. (2017) resort to a Picard iterative scheme, where each iteration is similar in nature to Figure 1 (or part of it, cf. Sect. 3.2), combined with a proxy approach for the $\mathrm{EC}_s(L)$ process, replaced by its unconditioned version as considered in Sect. 4.4. An unbiased approach to these equations would be another topic of future research.

APPENDIX. XVA NMC GPU Programming Optimization Techniques

Although XVA NMC computations seem naively suited to parallel architectures like graphic programming units (GPUs), a XVA NMC GPU implementation requires various computational complexity and memory storage optimizations, which are essential to achieve the potential ~ 100 speedup with respect to an optimized CPU implementation. These XVA NMC GPU programming optimization techniques are detailed in the remaining sections of the paper.

First, we need to provide some details on the GPU hardware/software aspects, referring the reader to Nvidia (2017b) for a detailed documentation. Originally de-

[e]Borrowing the terminology from An and Owen (2001).

veloped for gaming, beginning from around 2007, GPUs became extensively used for simulation. The most important specifications that make GPU architecture highly suited to NMC are (see Figure 3):

- A very large number of processing units grouped in streaming multiprocessors (SM). These units provide the sufficient computational power to process in parallel independent tasks;
- A large bandwidth to read/write data on the GPU random access memory (RAM) known as global memory. An important bandwidth is crucial to Monte Carlo since the performed operations are generally light when compared to the time spent to access to the memory;
- A cached memory space shared between (the 128) processors of the same streaming multiprocessor. This allows a very fast memory storage of arrays shared by various processors and makes possible the communication between them.

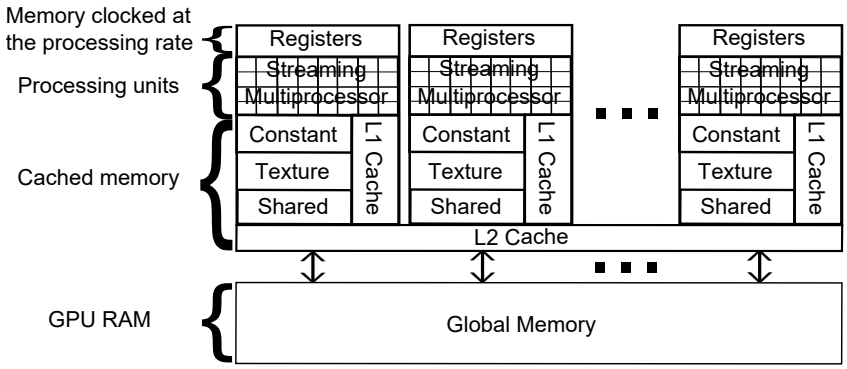

Fig. 3: A simple presentation of the Nvidia GPU architecture.

From the computational point of view, the streaming multiprocessors (SMs) execute blocks of (less than 1024) successive tasks known as threads. Once a block is associated to an SM, each streaming processor performs the required computations of various threads from the block. Within a block, the threads are organized in warps of 32 threads executed at the same time. Threads of the same warp are synchronized automatically at the hardware level, which is together convenient (in the sense that the programmer does not need care about it) and efficient. However, because of this intrinsic synchronization among threads of the same warp, the programmer has to ensure that the number of threads in a waiting state is as small as possible. Waiting states can be caused by conditional sentences like *if* clauses.

From the storage point of view, the global memory plays the role of the GPU RAM. Because it is the largest memory space on the GPU (from 1GB to 16GB), global memory is generally used by default. However, in order to reduce the time

of reading/writing values, instead of the global memory, one should **use registers and shared memory as much as possible**. As opposed to the global memory, shared memory and registers can only be accessed by their own streaming multiprocessors. Registers are the fastest and they are generally used for local variables. Although slower than registers, shared memory stores arrays that can be handled by all threads of the same block.

These are the main features that have made GPUs suitable, for more than a decade, to parallel computing and deep learning algorithms, resulting in their extensive use for artificial intelligence (see Coates, Huval, Wang, Wu, Catanzaro, and Andrew (2013)). We refer the reader to Nvidia (2017b) regarding further generic benefits of GPUs architecture in terms of concurrent execution, asynchronous data transfer between GPU and CPU, mapping the CPU memory, shuffles, tensor cores (introduced recently), etc..

There are many optimizations that must be considered in GPU programming but are not specific to NMC for XVA. The three most common ones consist in

- ensuring that access to the RAM is as coalescing as possible,
- reducing divergence in the code, and
- using the constant memory (cf. Figure 3), very fast as read-only, for model parameters.

RAM access coalescence and divergence management are as important as in a standard parallel implementation on CPU and we refer to Nvidia (2017b, 2017a) for further details. As for the constant memory requirement, it is not so important when the model is relatively simple so that the time spent in operations dominates the time spent to read the parameters. However, it becomes essential when the model involves a lot of parameters, e.g. with a local volatility model based on spline interpolation. In such cases, storing the parameters in a read-only memory considerably reduces the time spent to read the parameters and the execution time as a consequence.

In the developments that follow we mostly focus on the use of warps, registers, shared and global memories. For each optimization, we evaluate the speedup $= T_{\text{simp}}/T_{\text{opti}}$, where T_{opti} is the execution time of the optimized part of the code and T_{simp} is the execution time of the same part of the code but without the considered optimization.

Appendix A. Optimizations Related to the Time Grids Used in Factors Forwardations and Prices Backwardations

In this section, we present straightforward but very important optimizations needed for the nested simulation of any stochastic process on GPUs. Without loss of generality, let X be the univariate diffusion (local volatility model)

$$dX_t = X_t(b(t, X_t)dt + \sigma(t, X_t)dW_t),\ X_0 = x, \tag{A.1}$$

where W is a Brownian motion. To compute a derivative price Y driven by X, we first have to discretize several paths of the SDE (A.1) on a fine forwardation grid $\{0 = t_0, \ldots, t_{N_f} = T\}$. Then, given a backwardation subset $\{0 = s_0, \ldots, s_{N_b}\}$ of $\{t_0, \ldots, t_{N_f}\}$, one has to approximate

$$Y_{s_k} = \mathbb{E}\left(\phi(X_{s_{N_b}})|X_{s_k}\right), \text{ for every } 0 \leq k < N_b, \tag{A.2}$$

for a number of payoff functions ϕ.

In practice, the fine discretization $\{t_0, \ldots, t_{N_f}\}$ used for the forwardation of the underlying risk factors can be of the order of 100 time points per year in order to ensure an acceptable time discretization error to the underlying SDEs, whereas, in order to spare computational time and avoid an exploding accumulation of regression errors in case of FVA computations (or MtM computations for American claims), the coarse discretization $\{s_0, \ldots, s_{N_b}\}$ used for prices backwardation can be of the order of 10 time points per year. Hence the coarse discretization $\{s_0, \ldots, s_{N_b}\}$ is $N_f/N_b \sim 10$ times smaller than $\{t_0, \ldots, t_{N_f}\}$.

Be it for computing the European derivative price (A.2), its American version, or even risk measures conditional on the value of $X_{s_k}, 0 \leq k < N_b$, one has first to store various realizations $\{X_{s_k}\}_{0 \leq k < N_b}$ on the GPU RAM (or even on the CPU RAM if the GPU RAM is not sufficient). All the discretized intermediate realizations of $\{X_t\}_{t \in \{t_0, \ldots, t_{N_f}\} \setminus \{s_0, \ldots, s_{N_b}\}}$ should only be stored temporarily in the shared memory. We use shared memory that is faster than global memory. Morever, it is generally impossible to use only registers for this purpose, since one needs to store arrays of values in the case of multidimensional models. Registers, however, contain the state vector of the random number generator (RNG) used for simulating these temporary realizations.

More precisely, between each successive values $\{X_t\}_{t=t_k=s_{k'}, t_{k+1}=s_{k''}}$ stored in the global memory, one simulates temporarily $\{X_t\}_{t \in \{s_{k'+1}, \ldots, s_{k''-1}\}}$ stored in shared memory, which supposes the generation of various uniformly distributed random numbers stored in a register as the RNG state vector. Even though the RNG state is given by a vector, its size is fixed and thus does not have to be stored in an array. The RNG that we use is a CMRG which is a combination, presented in L'Ecuyer (1996), of two multiple recursive generators (MRGs). The first adaptation of CMRG to GPUs was proposed in Abbas-Turki et al. (2009, 2014).

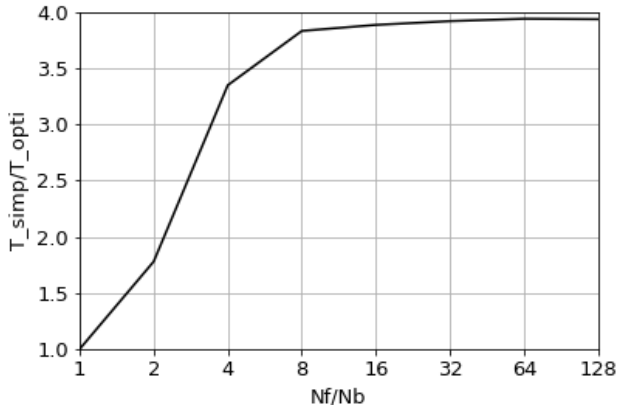

Fig. 4: The speedup, benefiting to all case studies (here in the context of the CVA and FVA case studies of Sect. 4.3.2), from using registers and shared memory during the nested simulation of the factor process X as a function of N_f/N_b, for N_f fixed to 128.

For $N_f = 128$, Figure 4 shows the speedup obtained from this strategy when $N_f/N_b = 1, \ldots, 128$ (i.e. $N_b = 128, \ldots, 1$). In the right part of Figure 4, the speedup tanks, since the application becomes less memory bound than compute bound. In the standard situation where $N_f/N_b \sim 10$, the speedup is generally greater than 3.5.

Appendix B. Optimizations Related to Default Times

Be it introduced by the XVA itself or by defaultable claims involved in the underlying market exposure, indicator functions on default events $\{\tau_i \in A\}$ increase the complexity of an efficient parallel implementation. For the first situation, presented in Section B.1, we aim at reducing the GPU memory occupation for limiting memory access and thus accelerating the execution. The optimization studied in Section B.2 shows how to reduce thread divergence in order to take advantage of the maximum computing power of GPUs. The situation faced in both cases is

$$\sum_{i \in \mathcal{S}} \mathbb{1}_{\tau_i \in A} Y^i, \tag{B.1}$$

where $\mathcal{S}$ is a set of obligors and τ_i can be simulated before Y^i. We keep using registers and shared memory as often as possible for both τ_i and Y^i simulations.

B.1. *Sorting Defaults for Multiple Counterparties*

Here $\mathcal{S}$ is a set of bank clients with associated counterparty exposures $\phi_i = \phi_i(t,x)$, the set A is a bounded time interval $(s_k, s_{k'}$ (with $k' > k$), and $Y^i = \phi_i(\tau_i, X_{\tau_i})$. Hence (B.1) can be rewritten as

$$\sum_{\text{bank clients}} \mathbb{1}_{s_k < \tau_i \leq s_{k'}} \, \phi_i(\tau_i, X_{\tau_i}). \tag{B.2}$$

If we had to simulate the process X on the fine discretization grid and then read the memory using τ_i, the quantity of memory needed would be quite significant, especially with NMC in view. Consequently, it is essential to only store in the global memory the values taken by the X_{τ_i} (such that $s_k < \tau_i \leq s_{k'}$), which can only be done after sorting the τ_i. This method spares the use of the global memory and reduces the number of writings to it. Due to the reduced number of writings, we also speed up the execution time, in particular as long as few defaults occurred, i.e., given the (low) intensity models that we use for the τ_i, during the first time steps of the algorithm: see the left panel in Figure 5.

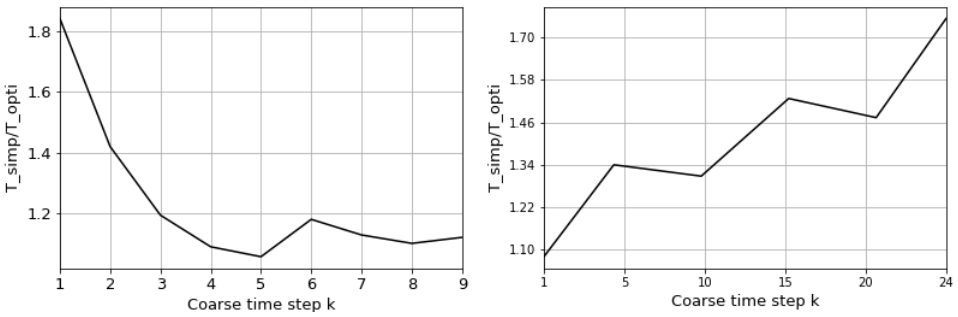

Fig. 5: The speedup obtained from using: Sorting defaults (left, in the context of the KVA case study of Sect. 4.4); Listing defaults (right, in the context of the CVA case studies of Sect. 4.2 and Sect. 4.3.1).

B.2. *Listing defaults for defaultable claims*

Now $\mathcal{S}$ is a set of reference obligors, $A = (s_k, \infty)$, and $Y^i = P_i(s_k, X_{s_k})$ for some coarse time s_k, where $P_i = P_i(t,x)$ is a suitable (pre-default) pricing function. The expression (B.1) reads as

$$\sum_{i \in \text{set of obigors}} \mathbb{1}_{\tau_i > s_k} \, P_i(s_k, X_{s_k}), \tag{B.3}$$

where $P_i(s_k, X_{s_k})$ is assumed more difficult to simulate than τ_i, e.g. because it requires another Monte Carlo simulation.

If we associate a thread to each realization of $\{\tau_i\}_{i\in\text{set of obligors}}$, then it is possible to have some threads returning for obligor $i = 1$ (say) $\mathbb{1}_{\tau_1>s_k} = 1$ and other threads from the same warp returning $\mathbb{1}_{\tau_1>s_k} = 0$. This situation creates thread divergence within the same warp, forcing threads/processors with $\mathbb{1}_{\tau_1>s_k} = 0$ to wait for the others.

To overcome this divergence, we list for each thread all the surviving obligors in order to make all threads computing together most of the time (there is always a small divergence due to the number of surviving obligors which is different between threads). Thus, the simulation of (B.3) is reduced to the simulation of

$$\sum_{i\in\text{list of surviving obigors}} P_i(s_k, X_{s_k}). \tag{B.4}$$

The corresponding speedup is demonstrated by the right panel in Figure 5. We see that the benefit of this strategy increases with the time index. This can be explained by the accumulated number of defaults that also increases with time, resulting in more divergence unless the above optimization is in place.

Appendix C. Optimized Sorting for $\mathbb{V}\mathrm{a}\mathbb{R}$ and $\mathbb{ES}$ Computations

Singh, Joshi, and Choudhary (2017) present a survey of the main GPU based sorting algorithms: radix sort, merge sort, sample sort and quick sort. However, sorting for computing risk measures such as $\mathbb{V}\mathrm{a}\mathbb{R}$ or $\mathbb{ES}$ is reduced to finding the $|\mathfrak{Q}|$ largest values from a list of $|\mathfrak{L}|$ values with $|\mathfrak{L}|\gg|\mathfrak{Q}|$, where $|\cdot|$ denotes the cardinality. This problem is less complex than sorting the whole list $\mathfrak{L}$ and it shares some similarities with the problem of finding the k nearest neighbours discussed, for instance, in Li and Amenta (2015). The latter paper is based on merge sort, with complexity $\mathrm{O}(|\mathfrak{L}|\log|\mathfrak{L}|)$, the parallelization of which is the most suited to GPUs (see Green, McColl, and Bader (2012)).

Here, we present an original method that allows to find the $|\widetilde{\mathfrak{L}}|$ largest values from a list of $|\mathfrak{L}|$ values in very few computations, of complexity $\mathrm{O}(|\mathfrak{L}|)$. The sublist $\widetilde{\mathfrak{L}}$ satisfies $|\mathfrak{Q}| \le |\widetilde{\mathfrak{L}}| \le 2|\mathfrak{Q}|\le|\mathfrak{L}|$. It is obtained from $\mathfrak{L}$ thanks to successive truncations that discard all the values smaller or equal to some threshold. One can then apply merge sort on $\widetilde{\mathfrak{L}}$ to get $\mathfrak{Q}$ with an overall complexity $\mathrm{O}(|\mathfrak{L}| + |\widetilde{\mathfrak{L}}|\log|\widetilde{\mathfrak{L}}|)$.

Denoting the median, mean, and standard deviation by μ, m, and dev, the main idea for obtaining at low cost $\widetilde{\mathfrak{L}}$ is based on Mallows inequality $|\mu - m| \le \text{dev}$, which allows using the (computationally cheap) m and dev instead of the (sort intensive) μ in order to pre-sort $\mathfrak{L}$: Basically (see Algorithm 2), starting from $\mathfrak{L}$, the list is truncated successively (4 or 5 times are typically enough in practice) by keeping the values superior or equal to $(m + \text{dev})$, m, or $(m - \text{dev})$. Because we only need to compute m and dev, which have a linear complexity, the complexity of pre-sorting $\mathfrak{L}$ to obtain $\widetilde{\mathfrak{L}}$ such that $|\mathfrak{Q}| \le |\widetilde{\mathfrak{L}}| \le 2|\mathfrak{Q}|$ is $\mathrm{O}(|\mathfrak{L}|)$.

Input: A large array $\mathfrak{L}$ of unordered values, the size $|\mathfrak{Q}|$ of $\mathfrak{Q}$;
$\widetilde{\mathfrak{L}} = \mathfrak{L}$;
while $|\widetilde{\mathfrak{L}}| > 2|\mathfrak{Q}|$ **do**
  Compute the mean m **and the standard deviation** dev **of** $\widetilde{\mathfrak{L}}$;
  if *the number of values in* $\widetilde{\mathfrak{L}}$ *that are bigger than* $m + \text{dev}$ *is* $\geq |\mathfrak{Q}|$ **then**
    New $\widetilde{\mathfrak{L}}$ **= values bigger than** $m + \text{dev}$ **of the previous** $\widetilde{\mathfrak{L}}$
  else if *the number of values in* $\widetilde{\mathfrak{L}}$ *bigger than* m *is* $\geq |\mathfrak{Q}|$ **then**
    New $\widetilde{\mathfrak{L}}$ **= values bigger than** m **of the previous** $\widetilde{\mathfrak{L}}$
  else if *the number of values in* $\widetilde{\mathfrak{L}}$ *bigger than* $m - \text{dev}$ *is* $\geq |\mathfrak{Q}|$ **then**
    New $\widetilde{\mathfrak{L}}$ **= values bigger than** $m - \text{dev}$ **of the previous** $\widetilde{\mathfrak{L}}$
Use merge sort on $\widetilde{\mathfrak{L}}$ **to get** $\mathfrak{Q}$
`Return(`*The array* $\mathfrak{Q}$.`)`

Algorithm 2: Optimized sorting algorithm for $\mathbb{V}a\mathbb{R}$ and $\mathbb{ES}$ computations.

When parallelized on GPUs, the operations presented in Algorithm 2 need be synchronized between threads. As we are performing NMC with a large number of different lists $\mathfrak{L}$ on which we need to compute some risk measure, we dedicate one warp of 32 threads to each list $\mathfrak{L}$. This solution is very convenient as threads of the same warp are automatically synchronized at the hardware level. Moreover, this approach is not memory intensive, hence it scales well and is not limited by the size of the shared memory as in Li and Amenta (2015).

Figure 6 shows the speedup obtained by this approach when compared to a straightforward implementation of complexity $O(|\mathfrak{Q}| \times |\mathfrak{L}|)$ for finding the $|\mathfrak{Q}|$ largest values among $|\mathfrak{L}|$ values.

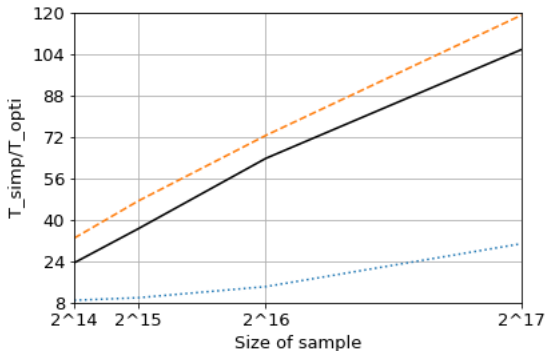

Fig. 6: The speedup obtained in the FVA case study of Sect. 4.4 by the truncation strategy for computing the '- -' 1% largest values, '—' 5% largest values, or '· · ·' 10% largest values from a sample list $\mathcal{L}$.

Appendix D. Regressions

Simulation/regression methods are widely used in mathematical finance. They occur in the context of Longstaff-Schwartz type dynamic programming (DP) algorithms that can be used for time 0 valuation of an early exercise derivative (see e.g. Abbas-Turki et al. (2014), Clément et al. (2002), Longstaff and Schwartz (2001)). Such algorithms can also be used for obtaining a proxy of the totality of a European pricing function (cf. Crépey and Rahal (2014)), but we emphasize that, as explained in Sect. 3.1, this comes without global error control.

Simulation/regression schemes are also commonly used for the resolution of non-linear BSDEs (see Gobet, Lemor, and Warin (2005)), such as the one that appears in asymmetric FVA computations. Other examples in the XVA context (subject to the same "local versus global error control" reservation as above though) include regression schemes that may be used for capturing conditional risk measures in the context of MVA or KVA computations (see e.g. Moran and Wilkens (2017), Green and Kenyon (2015), Anfuso et al. (2017), Anfuso et al. (2017)).

For a good overview on the kind of convergence associated to regression methods, we refer the reader to Gerhold (2011). As explained in Newey (1995), when the underlying risk factor process X has a density bounded away from zero (which can always be achieved numerically by truncation of X), the asymptotic number of trajectories has to be of the order of the cube of the cardinality of the regression basis, which we refer to as the cubic rule below.

D.1. *Regressions on Inner Trajectories*

When the XVA implementation requires a regression on inner NMC trajectories, this typically means relatively few paths as, on one hand, the computational cost duplicated on the inner nodes would be too large otherwise and, on the other hand, a non-negligible inner regression error is acceptable since it will be mitigated by outer averaging anyway.

The main ingredient relies on the resolution of a large[f] number of small[g] random symmetric linear systems. For instance (cf. Figure 7), in a simply nested setup, the implementation of a Longstaff-Schwartz algorithm on the inner trajectories requires $(N_b - k - 1)$ regressions at each coarse time $s_k, k = 1, \ldots, N_b - 1$, and for each outer trajectory $l \in \{1, \ldots, M_{(0)}\}$.

[f]Because of "inner" and of (even coarse) time stepping.
[g]Because few regressors are typically used in finance, especially here due to the cubic rule in regard of the (relatively small) number of (inner) trajectories.

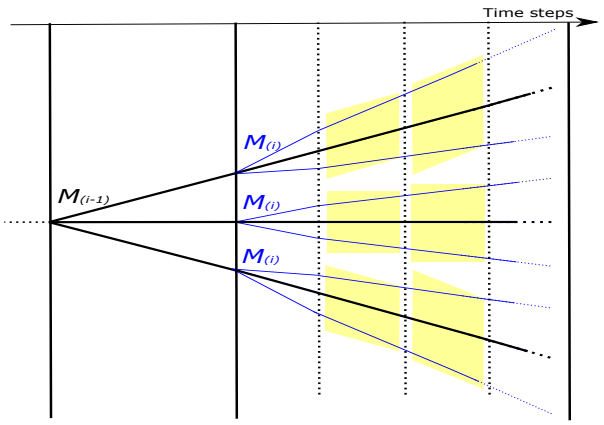

Fig. 7: Inner regression, such as the ones that appear in the Bermudan put CVA case study of Sect. 4.2, are symbolized by yellow pavings. The fine blue paths denote inner resimulated paths.

When the systems are well conditioned, the best solutions remain Cholesky or LDL^{T}[h] factorizations. A simple parallelization strategy would be to execute one resolution per thread. But this is far from optimal. For a system with n unknowns, the most appropriate GPU adaptation of LDL^{T}, presented in Abbas-Turki and Graillat (2017), involves $1 \leq n^* \leq n$ threads that work together to solve each system. In Figure 8 (Left), we see a clear benefit of using $n^* > 1$ threads per system when compared to the straightforward parallelization that involves only one thread per system. In most Longstaff-Schwartz applications, the number of regressors is smaller than 20 and the speedup is ~ 2.

In addition, Abbas-Turki and Graillat (2017) explain how to filter out the numerically null eigenvalues of an ill-conditioned regression covariance matrix, as the ones that arise in nested regressions based on very few inner trajectories.

D.2. *Regressions on Outer Trajectories*

In contrast to DP on inner trajectories, BSDE simulation on outer trajectories requires only few resolutions (one per coarse time step) of linear systems arising from pricing equations of the form

$$Y_{s_k} = \mathbb{E}\left(\Phi(Z_{s_{k+1}})|Z_{s_k}\right) \tag{D.1}$$

at coarse times s_k, where $Z = (X, H)$ includes both market factors X and default indicators encoded in the Markov chain component H (see Sect. 3.1). Many outer paths are required for the sake of accuracy, but this is doable without harm

[h]Cholesky implementation avoiding the use of square root functions.

provided the regression covariance matrices are estimated off-line based on a large number of outer trajectories. Once the covariance matrices computed, decomposed into eigenelements, and stored in the GPU RAM, they can be reused for any BSDE involving the same model on the outer layer of NMC.

Moreover, once we have the eigenelements of each covariance matrix, principal component analysis (PCA) can be used to reduce the dimensionality of the computations. Figure 8 (right) shows the speedup obtained from keeping only 90% to 95% of the spectrum (total variance) in a linear regression involving a common shock default model driven by three CIR processes ($|X| = 3$) and specifying the default of 100 obligors. As time passes, it becomes more difficult to explain most of the variance with few eigenvalues, because the structure of the likely default configurations becomes richer (as default intensities are "small"). However, in the 90% as in the 95% case, the dimensionality reduction speedup is never below 2, meaning that it is never necessary to use more than half of the eigenvalues, hence half of the time of what a full computation would take, to have a regression accounting for 90% or 95% of the spectrum.

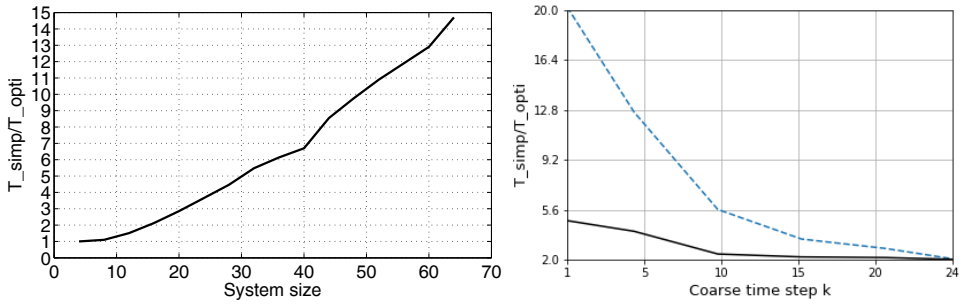

Fig. 8: The speedup obtained from optimized regressions. Left: in the case of regressions on inner trajectories, such as the ones that appear in the Bermudan put CVA case study of Sect. 4.2, by playing optimally with the number of threads n^* devoted to each linear system. Right: in the case of regressions on outer trajectories, such as the ones that appear in the FVA case study of Sect. 4.3.2, by limiting the number of eigenvalues accounted for in the regressions. '- - ' keeping 90% of the spectrum, '—' keeping 95% of the spectrum.

Acknowledgements

We thank Christophe Michel and Moez Mrad, from the quantitative CA-CIB research group, for their useful feedback, as well as the participants of the Deep Learning & Accélération GPU and RIO conferences (UPMC and and IMPA, November 2017), where this work was first presented, for useful questions and remarks. The

PhD thesis of Babacar Diallo is funded by a CIFRE grant from CA-CIB and French ANRT.

References

Abbas-Turki, L. and M. Mikou (2015). TVA on American Derivatives. hal-01142874 (working paper).

Abbas-Turki, L., S. Vialle, B. Lapreye, and P. Mercier (2009). *High Dimensional Pricing of Exotic European Contracts on a GPU Cluster, and Comparison to a CPU Cluster.* in IEEE IPDPS: Parallel and Distributed Computing in Finance.

Abbas-Turki, L. A., A. I. Bouselmi, and M. A. Mikou (2014). Toward a coherent Monte Carlo simulation of CVA. *Monte Carlo Methods and Applications 20*(3), 195–216.

Abbas-Turki, L. A. and S. Graillat (2017). Resolving small random symmetric linear systems on graphics processing units. *The Journal of Supercomputing 73*(4), 1360–1386.

Abbas-Turki, L. A., S. Vialle, B. Lapeyre, and P. Mercier (2014). Pricing derivatives on graphics processing units using Monte Carlo simulation. *Concurrency and Computation: Practice and Experience 26*(9), 1679–1697.

Albanese, C. and L. Andersen (2014). Accounting for OTC derivatives: Funding adjustments and the re-hypothecation option. Working paper available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2482955.

Albanese, C. and L. Andersen (2015). FVA: What's wrong, and how to fix it. *Risk Magazine*, September 54–56.

Albanese, C., L. Andersen, and S. Iabichino (2015). FVA: Accounting and risk management. *Risk Magazine*, February 64–68.

Albanese, C., T. Bellaj, G. Gimonet, and G. Pietronero (2011). Coherent global market simulations and securitization measures for counterparty credit risk. *Quantitative Finance 11*(1), 1–20.

Albanese, C., S. Caenazzo, and S. Crépey (2017). Credit, funding, margin, and capital valuation adjustments for bilateral portfolios. *Probability, Uncertainty and Quantitative Risk 2*(7), 26 pages. Available at http://rdcu.be/tHKo.

Albanese, C. and S. Crépey (2017). XVA analysis from the balance sheet. Working paper available at https://math.maths.univ-evry.fr/crepey.

Albanese, C. and H. Li (2010). Monte Carlo pricing using operator methods and measure changes. Working paper available at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=1484556.

An, J. and A. Owen (2001). Quasi-regression. *Journal of Complexity 17*(4), 588–607.

Andersen, L., D. Duffie, and Y. Song (2017). Funding value adjustments. First revised version available at https://ssrn.com/abstract=2746010.

Anfuso, F., D. Aziz, K. Loukopoulos, and P. Giltinan (2017). A sound modelling

and backtesting framework for forecasting initial margin requirements. *Risk Magazine*, May 1–6.

Armenti, Y. and S. Crépey (2017). XVA Metrics for CCP optimisation. Working paper available at https://math.maths.univ-evry.fr/crepey.

Bichuch, M., A. Capponi, and S. Sturm (2017). Arbitrage-free XVA. *Mathematical Finance*. First published online on 18 April 2017 (preprint version available at ssrn.2820257).

Brigo, D. and A. Pallavicini (2014). Nonlinear consistent valuation of CCP cleared or CSA bilateral trades with initial margins under credit, funding and wrong-way risks. *Journal of Financial Engineering 1*, 1–60.

Cesari, G., J. Aquilina, N. Charpillon, Z. Filipovic, G. Lee, and I. Manda (2010). *Modelling, Pricing, and Hedging Counterparty Credit Exposure*. Springer Finance.

Clément, E., D. Lamberton, and P. Protter (2002). An analysis of a least squares regression algorithm for American option pricing. *Finance and Stochastics 17*, 448–471.

Coates, A., B. Huval, T. Wang, D. Wu, B. Catanzaro, and N. Andrew (2013). Deep learning with COTS HPC systems. *Proceedings of the 30th International Conference on Machine Learning 28*(3), 1337–1345.

Crépey, S. (2013). *Financial Modeling: A Backward Stochastic Differential Equations Perspective*. Springer Finance Textbooks.

Crépey, S., T. R. Bielecki, and D. Brigo (2014). *Counterparty Risk and Funding: A Tale of Two Puzzles*. Chapman & Hall/CRC Financial Mathematics Series.

Crépey, S., R. Élie, W. Sabbagh, and S. Song (2017). When capital is a funding source: The XVA Anticipated BSDEs. Preprint version available at https://math.maths.univ-evry.fr/crepey.

Crépey, S., G. Fort, E. Gobet, and U. Stazhynski (2017). Uncertainty quantification for stochastic approximation limits using chaos expansion. hal-01629952.

Crépey, S. and T. M. Nguyen (2016). Nonlinear Monte Carlo schemes for counterparty risk on credit derivatives. In *Challenges in Derivatives Markets*, Springer Proceedings in Mathematics, pp. 53–82. Springer.

Crépey, S. and A. Rahal (2014). Simulation/regression pricing schemes for CVA computations on CDO tranches. *Communications in Statistics–Theory and Methods 43*(7), 1390–1408.

Crépey, S. and S. Song (2016). Counterparty risk and funding: Immersion and beyond. *Finance and Stochastics 20*(4), 901–930.

Delmas, J.-F. and B. Jourdain (2006). Modèles aléatoires. Volume 57 of *Mathématiques et Applications*. Springer.

Elouerkhaoui, Y. (2007). Pricing and hedging in a dynamic credit model. *International Journal of Theoretical and Applied Finance 10*(4), 703–731.

Elouerkhaoui, Y. (2016). From FVA to KVA: including cost of capital in derivatives pricing. *Risk Magazine*, March 70–75.

Elouerkhaoui, Y. (2017). *Credit Correlation: Theory and Practice*. Palgrave

Macmillan. Forthcoming.

Embrechts, P., C. Klueppelberg, and T. Mikosch (1997). Modelling extremal events for insurance and finance. Volume 33 of *Applications of Mathematics*. Springer.

Fujii, M. and A. Takahashi (2012a). Analytical approximation for non-linear FBSDEs with perturbation scheme. *International Journal of Theoretical and Applied Finance 15*(5), 1250034 (24 pages).

Fujii, M. and A. Takahashi (2012b). Perturbative expansion of FBSDE in an incomplete market with stochastic volatility. *Quarterly Journal of Finance 2*(3), 1250015 (24 pages).

Gerhold, S. (2011). The Longstaff-Schwartz algorithm for Lévy models: Results on fast and slow convergence. *The Annals of Applied Probability 21*(2), 589–608.

Giles, M. (2008). Multilevel Monte Carlo path simulation. *Operations Research 56*, 607–617.

Gobet, E., J.-P. Lemor, and X. Warin (2005). A regression-based Monte Carlo method to solve backward stochastic differential equations. *The Annals of Applied Probability 15*(3), 2172–2202.

Gobet, E. and S. Pagliarani (2015). Analytical approximations of BSDEs with nonsmooth driver. *SIAM Journal on Financial Mathematics 6*, 919–958.

Gordy, M. B. and S. Juneja (2010). Nested simulation in portfolio risk measurement. *Management Science 56*(10), 1833–1848.

Green, A. (2015). *XVA: Credit, Funding and Capital Valuation Adjustments*. Wiley.

Green, A. and C. Kenyon (2015). MVA by replication and regression. *Risk Magazine*, May 82–87.

Green, A., C. Kenyon, and C. Dennis (2014). KVA: capital valuation adjustment by replication. *Risk Magazine*, December 82–87. Preprint version "KVA: capital valuation adjustment" available at ssrn.2400324.

Green, O., R. McColl, and D. A. Bader (2012). GPU Merge Path – A GPU Merging Algorithm. In *26th ACM International Conference on Supercomputing (ICS)*, pp. 25–29. The ACM Digital Library.

Henry-Labordère, P. (2012). Cutting CVA's complexity. *Risk Magazine*, July 67–73.

Hull, J. and A. White (2012). CVA and wrong way risk. *Financial Analyst Journal 68*, 58–69.

Iben Taarit, M. (2017). *Pricing of XVA Adjustments: from Expected Exposures to Wrong-Way risks*. Ph. D. thesis, Université Paris-Est. Forthcoming.

L'Ecuyer, P. (1996). Combined multiple recursive random number generators. *Operations Research 44*, 5.

Li, M. and F. Mercurio (2015). Jumping with default: wrong-way risk modelling for CVA. *Risk Magazine*, November.

Li, S. and N. Amenta (2015). Brute-force k-nearest neighbors search on the GPU.

In *Similarity Search and Applications*, pp. 259–270. Springer.

Longstaff, F. A. and E. S. Schwartz (2001). Valuing American options by simulation: A simple least-squares approach. *The Review of Financial Studies 14* (1), 113–147.

Moran, L. and S. Wilkens (2017). Capturing initial margin in counterparty risk calculations. *Journal of Risk Management in Financial Institutions 10*, 118–129.

Newey, W. K. (1995). Convergence rates and asymptotic normality for series estimators. *Journal of Econometrics 79*, 147–168.

Nvidia (2017a). Cuda C best practices guide.

Nvidia (2017b). Cuda C programming guide.

Nvidia (2017c). Nvidia NVLink TM high-speed interconnect: Application performance.

Pykhtin, M. (2012). General wrong-way risk and stress calibration of exposure. *Journal of Risk Management in Financial Institution 5*, 234–251.

Reghai, A., O. Kettani, and M. Messaoud (2015). CVA with Greeks and AAD. *Risk Magazine*, December.

Singh, D. P., I. Joshi, and J. Choudhary (2017). Survey of GPU based sorting algorithms. *International Journal of Parallel Programming*, DOI 10.1007/s10766–017–0502–5.