DEEP XVA SOLVER – A NEURAL NETWORK BASED COUNTERPARTY CREDIT RISK MANAGEMENT FRAMEWORK

ALESSANDRO GNOATTO, ATHENA PICARELLI, AND CHRISTOPH REISINGER

November 29, 2022

ABSTRACT. In this paper, we present a novel computational framework for portfolio-wide risk management problems, where the presence of a potentially large number of risk factors makes traditional numerical techniques ineffective. The new method utilises a coupled system of BSDEs for the valuation adjustments (xVA) and solves these by a recursive application of a neural network based BSDE solver. This not only makes the computation of xVA for high-dimensional problems feasible, but also produces hedge ratios and dynamic risk measures for xVA, and allows simulations of the collateral account.

1. INTRODUCTION

As a consequence of the 2007–2009 financial crisis, academics and practitioners have been redefining and augmenting key concepts of risk management. This made it necessary to reconsider many widely used methodologies in quantitative and computational finance.

It is now generally accepted that a reliable valuation of a financial product should account for the possibility of default of any agent involved in the transaction. Moreover, the trading activity is nowadays funded by resorting to different sources of liquidity (the interest rate multi-curve phenomenon; see, e.g., Cuchiero et al. (2019)), so that the existence of a single funding stream with a unique risk-free interest rate no longer represents a realistic assumption. Additionally, the increasingly important role of collateral agreements demands for a portfolio-wide view of valuation.

These stylized facts are incorporated into the valuation equations through value adjustments (xVA). Value adjustments are terms to be added to, or subtracted from, an idealized reference portfolio value, computed in the absence of frictions, in order to obtain the final value of the transaction.

The literature on counterparty credit risk and funding is large and we only attempt to provide insights on the main references as they relate to our work. Possibly the first contribution on the subject is a model for credit risk asymmetry in swap contracts in Duffie and Huang (1996). Before the 2007–2009 financial crisis, we have the works of Brigo and Masetti (2005) and Cherubini (2005), where the concept of credit valuation adjustment (CVA) is analyzed. The possibility of default of both counterparties involved in the transaction, represented by the introduction of the debt valuation adjustment (DVA), is investigated, among others, in Brigo et al. (2011, 2014).

Another important source of concern to practitioners apart from default risk is represented by funding costs. A parallel stream of literature emerged during and after the financial crisis to generalize valuation equations in the presence of collateralization agreements. In a Black-Scholes economy, Piterbarg (2010) gives valuation formulas both in the collateralized and uncollateralized case. Generalizations to the case of a multi-currency economy can be found in Piterbarg (2012), Fujii et al. (2010, 2011), and

Gnoatto and Seiffert (2021). The funding valuation adjustment (FVA) is derived under alternative assumptions on the Credit Support Annex (CSA) in Pallavicini et al. (2011), while Brigo and Pallavicini (2014) also discusses the role of central counterparties for funding costs. A general approach to funding in a semimartingale setting is provided by Bielecki and Rutkowski (2015).

Funding and default risk need to be united in a single risk management framework to account for all possible frictions and their interplay. Contributions in this sense can be found in Brigo et al. (2018) by means of the so-called discounting approach. Burgard and Kjaer (2011a,b) generalize the classical Black-Scholes replication approach to include some of the aforementioned effects. A more general backward stochastic differential equation (BSDE) approach is provided by Crépey (2015a,b) and Bichuch et al. (2018a,b). The equivalence between the discounting approach and the BSDE-based replication approaches is demonstrated in Brigo et al. (2018).

A common fundamental feature of such generalized risk management frameworks is the necessity to adopt a portfolio-wide point of view in order to properly account for risk mitigation benefits arising from diversified positions. Adopting such portfolio-wide models, as is the present market practice in financial institutions, involves high-dimensional joint simulations of all positions within a portfolio.

Commonly used numerical techniques (see for instance Shöftner (2008); Karlsson et al. (2016); Broadie et al. (2015); Joshi and Kwon (2016)) make use of regression approaches, based on a modification of the Least-Squares Monte Carlo approach in Longstaff and Schwartz (2001), to alleviate the high computational cost of fully nested Monte Carlo simulations such as those initially proposed in Gordy and Juneja (2010); Broadie et al. (2011). We refer to Albanese et al. (2017) for a high-performance GPU implementation of nested Monte Carlo for bilateral xVA computations in a modern set-up including credit, margin and capital, for a large book of about 200,000 trades with 2000 counterparties. For an application of adjoint algorithmic differentiation (AAD) to xVA simulation by regression see, for instance, Capriotti et al. (2017); Fries (2019).

An alternative, hybrid, approach to counterparty risk computations is taken in de Graaf et al. (2014), where standard pricing methods are applied to the products in the portfolio and outer Monte Carlo estimators for exposures. Techniques based purely on PDEs generally suffer from the *curse of dimensionality*, a rapid increase of computational cost in presence of high dimensional problems. A PDE approach with factor-based dimension reduction has been proposed in de Graaf et al. (2018).

In the broader context of high-dimensional problems involving large amounts of data, machine learning techniques have witnessed dramatically increasing popularity. Of particular interest is the concept of an artificial neural network (ANN). From a mathematical perspective, ANNs are multiple nested compositions of relatively simple multivariate functions. The term deep neural networks refers to ANNs with several interconnected layers. One remarkable property of ANNs is given in the 'Universal Approximation Theorem', which essentially states that any continuous function in any dimension can be represented to arbitrary accuracy by means of an ANN, and has been proven in different versions, starting from the remarkable insight of Kolmogorov's Representation Theorem in Kolmogorov (1956) and the seminal works of Cybenko (1989) and Hornik (1991). Recently, building heavily on earlier work of Jentzen et al. (2018), the recent results by Reisinger and Zhang (2020) have proven that deep ANNs can overcome the curse of dimensionality for approximating (nonsmooth) solutions of partial differential equations arising from (open-loop control of) SDEs. A result to the same effect has been shown for heat equations with a zero-order nonlinearity in Hutzenthaler et al. (2018). This is potentially useful in the context of risk management as simple models for CVA can be expressed in this form. For a recent literature survey of applications of neural networks to pricing, hedging and risk management problems more generally we refer the reader to Ruf and Wang (2019).

In this paper, we investigate the application of ANNs to solve high-dimensional BSDEs arising from risk management problems. Indeed, in the classical continuous-time mathematical finance literature the random behavior of the simple financial assets composing a portfolio is typically described by means of multi-dimensional Brownian motions and forward stochastic differential equations (SDEs). In this setting, BSDEs naturally arise as a representation of the evolution of the hedging portfolio, where the terminal condition represents the target payoff (see, e.g., El Karoui et al. (1997)). In essence, (numerically) solving a BSDE is equivalent to identifying a risk management strategy.

Numerical BSDE methods published recently for xVA computations for single derivatives include Borovykh et al. (2018). The difficulty of extending these computational techniques to the portfolio setting is alluded to in Remark 11 of Ninomiya and Shinozaki (2019).

Here, we will consider a discretized version of the BSDE and parametrize the (high dimensional) control (i.e., hedging) process at every point in time by means of a family of ANNs. Once written in this form, BSDEs can be viewed as model-based reinforcement learning problems. The ANN parameters are then fitted so as to minimize a prescribed loss function.

The line of computational methods we follow has been initiated in the context of high-dimensional nonlinear PDEs in Han et al. (2018) and further investigated in Han and Long (2020) and Fujii et al. (2019), and has led to a class of methods for the solution of BSDEs (characterised by parametrisation of the Markovian control by ANNs), which we will collectively refer to as the Deep BSDE Solver for simplicity. By way of financial applications, and xVA specifically, a primal-dual extension to the Deep BSDE Solver has been developed in Henry-Labordere (2017) and tested on stylised CVA- and IM(Initial Margin)-type PDEs; the Deep BSDE Solver has also been applied specifically to exposure computations for a Bermudan swaption and a cross-currency swap in She and Grecu (2017).

Our approach goes beyond these earlier works in the following regards: we

- consider a rigorous, generic BSDE model for the dynamics of xVA, including CVA, DVA, FVA and ColVA (collateral valuation adjustment), for a derivative portfolio;
- introduce algorithms for the computation of 'non-recursive' xVAs – such as CVA and DVA – and 'recursive' xVAs – such as FVA – by (recursive) application of the Deep BSDE Solver, and deduce *a posteriori* bounds on the error of the neural network approximations;
- show how the method can be used for the simulation of xVA sensitivities and collateral, and provide careful numerical tests, showing good (i.e., basis point) accuracy for different adjustment computations, including an example with 100 underlying assets.

We will refer to our method as *Deep xVA Solver*. More recently, conditional risk measure computations (VaR and ES), based on deep learning regression, have been proposed in an xVA framework in Albanese et al. (2021), using a similar numerical approach to the one developed independently for BSDEs in Huré et al. (2020). Different from Han et al. (2018), this solver approximates the value function, not the control, by means of an ANN and reconstructs it at each time step by dynamic programming techniques. A comparison of the performance and robustness of the two approaches will require comprehensive testing in industry-relevant settings. We see as a structural advantage of our algorithm that it allows to obtain the xVA hedging strategy with no need of further computation (i.e. differentiation).

The applicability of the presented methodology is largely independent of the particular choice of the xVA framework. In particular, we do not take a position in the so-called *FVA debate* or on the question of including KVA in the pricing equation. The term *FVA debate* here refers to the possible overlap between the debt value adjustment (DVA) and the funding benefit adjustment (FBA). This overlap has been addressed in Brigo et al. (2019). The inclusion of KVA is still debated, noting, for

example, the recent criticism of KVA in Andersen et al. (2019). Our approach is general enough to accommodate different specifications of the price decomposition. In particular, our methodology can be applied immediately to the framework of Brigo et al. (2019).

We restrict the presentation of the method to a single counter-party – or 'netting set' – for simplicity, as is routinely done in banks. There are economic grounds for extending the computation to multiple netting sets simultaneously (see, e.g., Albanese et al. (2021)) and our method generalises accordingly. The paper is organized as follows. The financial framework is established in Section 2. In Section 3, after shortly recalling the main features of the Deep BSDE Solver presented in Han et al. (2018), the algorithm for xVA computation is introduced. Numerical results for a selection of test cases are shown in Section 4, while Section 6 concludes.

2. The financial market

For concreteness, we adopt the market setup of Biagini et al. (2021) and subsequently formulate our computational methods in the context of this model. Let us re-iterate the point elaborated in the introduction, however, that the computational framework, which is the focus of this article, is adaptable to a range of model specifications.

We fix a time horizon $T < \infty$ for the trading activity of two agents named the *bank* (B) and the *counterparty* (C). Unless otherwise stated, throughout the paper we assume the bank's perspective and refer to the bank as the *hedger*.

All underlying processes are modeled over a probability space $(\Omega, \mathcal{G}, \mathbb{G}, \mathbb{Q})$, where $\mathbb{G} = (\mathcal{G}_t)_{t \in [0,T]} \subseteq \mathcal{G}$ is a filtration satisfying the usual assumptions ($\mathcal{G}_0$ is assumed to be trivial). We denote by τ^B and τ^C the *time of default* of the bank and the counterparty, respectively. Specifically, we assume that $\mathbb{G} = \mathbb{F} \vee \mathbb{H}$, where $\mathbb{F} = (\mathcal{F}_t)_{t \in [0,T]}$ is a reference filtration satisfying the usual assumptions and $\mathbb{H} = \mathbb{H}^B \vee \mathbb{H}^C$, with $\mathbb{H}^j = (\mathcal{H}_t^j)_{t \in [0,T]}$ for $\mathcal{H}_t^j = \sigma(H_u \,|\, u \le t)$, and $H_t^j := \mathbb{1}_{\{\tau^j \le t\}}$, $j \in \{B, C\}$. We set

$$\tau = \tau^C \wedge \tau^B.$$

In the present paper we will extensively make use of the so called *Immersion Hypothesis* (see, e.g., Bielecki and Rutkowski (2004)).

Assumption 1. *Any local $(\mathbb{F}, \mathbb{Q})$-martingale is a local $(\mathbb{G}, \mathbb{Q})$-martingale.*

We consider the following spaces:

- $L^2(\mathbb{R}^d)$ is the space of all $\mathcal{F}_T$-measurable $\mathbb{R}^d$-valued random variables $X : \Omega \mapsto \mathbb{R}^d$ such that $\|X\|^2 = \mathbb{E}\left[|X|^2\right] < \infty$.
- $\mathbb{H}^{2,q \times d}$ is the space of all predictable $\mathbb{R}^{q \times d}$-valued processes $\phi : \Omega \times [0,T] \mapsto \mathbb{R}^{q \times d}$ such that $\mathbb{E}\left[\int_0^T |\phi_t|^2 dt\right] < \infty$.
- $\mathbb{S}^2$ the space of all adapted processes $\phi : \Omega \times [0,T] \mapsto \mathbb{R}^{q \times d}$ such that $\mathbb{E}\left[\sup_{0 \le t \le T} |\phi_t|^2\right] < \infty$.

2.1. Basic traded assets.

Risky assets. For $d \ge 1$, we denote by S^i, $i = 1, \ldots, d$, the *ex-dividend price* (i.e. the price) of risky securities. All S^i are assumed to be càdlàg $\mathbb{F}$-semimartingales.

Let $W^{\mathbb{Q}} = (W_t^{\mathbb{Q}})_{t \in [0,T]}$ be a d-dimensional $(\mathbb{F}, \mathbb{Q})$-Brownian motion (hence a $(\mathbb{G}, \mathbb{Q})$-Brownian motion, thanks to Assumption 1). We introduce the coefficient functions $\mu : \mathbb{R}_+ \times \mathbb{R}^d \mapsto \mathbb{R}^d$, $\sigma : \mathbb{R}_+ \times \mathbb{R}^d \mapsto \mathbb{R}^{d \times d}$, which are assumed to satisfy standard conditions ensuring existence and uniqueness of strong

solutions of SDEs driven by the Brownian motion $W^{\mathbb{Q}}$. We assume that

(2.1)
$$\begin{cases} \mathrm{d}S_t = \mu(t, S_t)\,\mathrm{d}t + \sigma(t, S_t)\,\mathrm{d}W_t^{\mathbb{Q}}, \\ S_0 = s_0 \in \mathbb{R}^d, \end{cases}$$

on $[0, T]$, where $S_t = (S_t^1, \ldots, S_t^d) \in \mathbb{R}^d$. Note that we are not postulating that the processes S^i are positive.

Throughout the paper we assume that the market is complete for the sake of simplicity.

Cash accounts. Given a stochastic return process $x := (x_t)_{t \geq 0}$, which is assumed bounded, right-continuous and $\mathbb{F}$-adapted, we define the cash account B^x with unitary value at time 0, as the strictly positive continuous processes of finite variation

(2.2)
$$B_t^x := \exp\left\{ \int_0^t x_s\,\mathrm{d}s \right\}, \, t \in [0, T].$$

In particular, $B^x := (B_t^x)_{t \in [0,T]}$ is also continuous and adapted.

Defaultable bonds. Default times are assumed to be exponentially distributed random variables with time-dependent intensity

$$\Gamma_t^j = \int_0^t \lambda_s^{j,\mathbb{Q}}\,\mathrm{d}s, \quad t \in [0, T], \ j \in \{B, C\},$$

where $\lambda^{j,\mathbb{Q}}$ are non-negative bounded processes.

We introduce two risky bonds with maturity $T^\star \leq T$ issued by the bank and the counterparty. We directly state their dynamics under $\mathbb{Q}$. We refer to Biagini et al. (2021) for more details. The risky bonds evolve according to

(2.3)
$$\mathrm{d}P_t^j = r_t^j P_t^j\,\mathrm{d}t - P_{t-}^j\,dM_t^{j,\mathbb{Q}}, \quad j \in \{B, C\},$$

where $M^{j,\mathbb{Q}}$, $j \in \{B, C\}$ are compensated Poisson random measures, see equation (3.6) in Biagini et al. (2021).

2.2. **xVA framework.** We consider a family of contingent claims within a portfolio with agreed dividend stream $A^m = (A_t^m)_{t \in [0,T]}$, $m = 1, \ldots, M$, and set $\bar{A}_t^m := \mathbb{1}_{\{t < \tau\}} A_t^m + \mathbb{1}_{\{t \geq \tau\}} A_{\tau-}^m$. We let $T_m \leq T$ denote the maturity time of the m-th contract. The value of the single claim within the portfolio, ignoring any counterparty risk or funding issue, that we refer to as *clean values*, are denoted by $(\widehat{V}_t^m)_{m=1,\ldots,M}$ and satisfy the following forward-backward stochastic differential equations (FBSDEs), for $m = 1, \ldots, M$,

(2.4)
$$\begin{cases} -\mathrm{d}\widehat{V}_t^m = \mathrm{d}A_t^m - r_t\widehat{V}_t^m\,\mathrm{d}t - \sum_{k=1}^d \widehat{Z}_t^{m,k}\,\mathrm{d}W_t^{k,\mathbb{Q}}, \\ \widehat{V}_{T_m}^m = 0, \end{cases}$$

which reads, in integral form,

(2.5)
$$\widehat{V}_t^m := \mathbb{E}^{\mathbb{Q}}\left[B_t^r \int_{(t, T_m]} \frac{\mathrm{d}A_u^m}{B_u^r} \,\middle|\, \mathcal{F}_t \right], \quad t \in [0, T_m],$$

where r is a collateral rate in an idealized perfect collateral agreement.

For simplicity, we restrict ourselves to Europan-type contracts and, for $m = 1, \ldots M$, denoted by the Lipschitz function g_m the payoff of the option we write $A_t^m = \mathbb{1}_{\{t \geq T_m\}} g_m(S_{T_m})$. In this case, with an

abuse of notation, instead of equation (2.4) we consider

$$(2.6) \qquad \begin{cases} -\mathrm{d}\widehat{V}_t^m = -r_t\widehat{V}_t^m \,\mathrm{d}t - \sum_{k=1}^d \widehat{Z}_t^{m,k} \,\mathrm{d}W_t^{k,\mathbb{Q}}, \\ \widehat{V}_{T_m}^m = g_m(S_{T_m}). \end{cases}$$

Observe that the system (2.1) and (2.6) is decoupled, in the sense that the forward equation (2.1) does not exhibit a dependence on the backward component.

We continue to follow the framework of Biagini et al. (2021), where the portfolio dynamics are stated in the form of a BSDE under the enlarged filtration $\mathbb{G}$. We set

$$(2.7\text{a}) \qquad Z_t^k := \sum_{i=1}^d \xi_t^i \sigma^{i,k}(t, S_t), \qquad\qquad\qquad k = 1, \dots, d,$$

$$(2.7\text{b}) \qquad U_t^j := -\xi_t^j P_{t-}^j, \qquad\qquad\qquad j \in \{B, C\},$$

$$(2.7\text{c}) \qquad f(t, V, C) := -\Big[(r_t^{f,l} - r_t)(V_t - C_t)^+ - (r_t^{f,b} - r_t)(V_t - C_t)^- \\ + (r_t^{c,l} - r_t)C_t^+ - (r_t^{c,b} - r_t)C_t^- \Big],$$

where

- ξ^i, $i = 1, \dots, d$, are the positions in risky assets, while ξ^B, ξ^C are the position in the bank and counterparty bond respectively;
- $r^{f,l}, r^{f,b}$ represent unsecured funding lending and borrowing rates;
- $r^{c,l}, r^{c,b}$ denote the interest on posted and received variation margin (collateral);
- $\sigma^{i,k}(t, S_t)$ is the (i,k)-th entry of the matrix $\sigma(t, S_t)$, for $i, k = 1, \dots, d$;
- C^+ and C^- represent the posted and received variation margin/collateral and $C = C^+ - C^-$.

All above processes are assumed to satisfy suitable regularity conditions ensuring existence and uniqueness for a solution to BSDE (2.8) below. Both posted and received collateral are assumed to be Lipschitz functions of the clean value of the derivative portfolio and we will write $C_t = C(V_t)$.

We denote by V the *full contract* value, i.e. the portfolio value including counterparty risk and multiple curves. The $\mathbb{G}$-BSDE for the portfolio's dynamics then has the form on $\{t < \tau\}$

$$(2.8) \qquad \begin{cases} -\mathrm{d}V_t = \sum_{m=1}^M \mathrm{d}\bar{A}_t^m + (f(t, V, C) - r_t V_t)\,\mathrm{d}t - \sum_{k=1}^d Z_t^k \,\mathrm{d}W_t^{k,\mathbb{Q}} - \sum_{j \in \{B,C\}} U_t^j \,\mathrm{d}M_t^{j,\mathbb{Q}}, \\ V_\tau = \theta_\tau(\widehat{V}, C), \qquad \text{with} \\ \theta_\tau(\widehat{V}, C) := \widehat{V}_\tau + \mathbb{1}_{\{\tau^C < \tau^B\}}(1 - R^C)\left(\widehat{V}_\tau - C_{\tau-}\right)^- - \mathbb{1}_{\{\tau^B < \tau^C\}}(1 - R^B)\left(\widehat{V}_\tau - C_{\tau-}\right)^+, \end{cases}$$

where $\widehat{V}_t := \sum_{m=1}^M \widehat{V}_t^m$ and R^B, R^C are two positive constants representing the recovery rate of the bank and the counterparty, respectively.

In their Theorem 3.16, Biagini et al. (2021) show that there exists a unique solution (V, Z, U) for the $\mathbb{G}$-BSDE (2.8), and the process V assumes the following form on $\{t < \tau\}$:

$$(2.9) \qquad V_t = B_t^r \mathbb{E}^{\mathbb{Q}}\left[\sum_{m=1}^M \int_{(t,\tau\wedge T]} \frac{\mathrm{d}\bar{A}_u^m}{B_u^r} + \int_t^{\tau\wedge T} \frac{f(u, V, C)}{B_u^r} \,\mathrm{d}u + \mathbb{1}_{\{\tau \leq T\}} \frac{\theta_\tau(\widehat{V}, C)}{B_\tau^r} \,\bigg|\, \mathcal{G}_t \right].$$

To prove existence and uniqueness for the $\mathbb{G}$-BSDE, Biagini et al. (2021) employ the technique introduced by Crépey (2015a) and reformulate the problem under the reduced filtration $\mathbb{F}$. Stated in such a form, the problem is also more amenable to numerical computations.

We consider the following $\mathbb{F}$-BSDE on $[0, T]$:

$$(2.10) \qquad \begin{cases} -\mathrm{d}\overline{\mathrm{XVA}}_t = \bar{f}(t, \widehat{V}_t, \overline{\mathrm{XVA}}_t)\, \mathrm{d}t - \sum_{k=1}^{d} \overline{Z}_t^k\, \mathrm{d}W_t^{k,\mathbb{Q}}, \\ \overline{\mathrm{XVA}}_T = 0, \end{cases}$$

where

$$
\begin{aligned}
(2.11) \qquad \bar{f}(t, \widehat{V}_t, \overline{\mathrm{XVA}}_t) := {} & -(1 - R^C)\left(\widehat{V}_t - C_t\right)^- \lambda_t^{C,\mathbb{Q}} \\
& + (1 - R^B)\left(\widehat{V}_t - C_t\right)^+ \lambda_t^{B,\mathbb{Q}} \\
& + (r_t^{f,l} - r_t)\left(\widehat{V}_t - \overline{\mathrm{XVA}}_t - C_t\right)^+ - (r_t^{f,b} - r_t)\left(\widehat{V}_t - \overline{\mathrm{XVA}}_t - C_t\right)^- \\
& + (r_t^{c,l} - r_t)C_t^+ - (r_t^{c,b} - r_t)C_t^- - (r_t + \lambda_t^{C,\mathbb{Q}} + \lambda_t^{B,\mathbb{Q}})\overline{\mathrm{XVA}}_t.
\end{aligned}
$$

By standard results on BSDEs, see e.g. Delong (2017, Theorem 4.1.3, Theorem 3.1.1), the existence and uniqueness of solutions $(\widehat{V}^m, \widehat{Z}^m) \in \mathbb{S}^2(\mathbb{R}) \times \mathbb{H}^{2,q \times 1}$, for $m = 1, \ldots, M$, and $(\overline{\mathrm{XVA}}, \overline{Z}) \in \mathbb{S}^2(\mathbb{R}) \times \mathbb{H}^{2,q \times 1}$ to, respectively, (2.6) and (2.10), holds under the following conditions:

$$r^{f,l}, r^{f,b}, r^{c,l}, r^{c,b}, r, \lambda^{B,\mathbb{Q}}, \lambda^{C,\mathbb{Q}} \text{ are bounded processes;}$$

$$|\mu(t, x) - \mu(t, x')| + |\sigma(t, x) - \sigma(t, x')| \leq C|x - x'|,$$

$$|\sigma(t, x)| + |\mu(t, x)| \leq C(1 + |x|).$$

The process $\overline{\mathrm{XVA}}$ coincides with the *pre-default* xVA process. Indeed, given the pre-default value process $\overline{V}$ such that $\overline{V}_t \mathbb{1}_{\{t < \tau\}} = V_t \mathbb{1}_{\{t < \tau\}}$, on $\{t < \tau\}$ the solution to (2.8) can be represented as

$$\overline{V}_t = \widehat{V}_t - \overline{\mathrm{XVA}}_t.$$

Moreover, defining the process $\tilde{r} = (\tilde{r}_t)_{t \in [0,T]}$ as $\tilde{r} := r + \lambda^{C,\mathbb{Q}} + \lambda^{B,\mathbb{Q}}$, it has been shown in Biagini et al. (2021, Corollary 3.17) that the process $\overline{\mathrm{XVA}}$ admits the representation

$$(2.12) \qquad \overline{\mathrm{XVA}}_t = -\overline{\mathrm{CVA}}_t + \overline{\mathrm{DVA}}_t + \overline{\mathrm{FVA}}_t + \overline{\mathrm{ColVA}}_t,$$

where

$$(2.13) \qquad \overline{\mathrm{CVA}}_t := B_t^{\tilde{r}} \mathbb{E}^{\mathbb{Q}}\left[(1 - R^C) \int_t^T \frac{1}{B_u^{\tilde{r}}} \left(\widehat{V}_u - C_u\right)^- \lambda_u^{C,\mathbb{Q}}\, \mathrm{d}u \,\middle|\, \mathcal{F}_t \right],$$

$$(2.14) \qquad \overline{\mathrm{DVA}}_t := B_t^{\tilde{r}} \mathbb{E}^{\mathbb{Q}}\left[(1 - R^B) \int_t^T \frac{1}{B_u^{\tilde{r}}} \left(\widehat{V}_u - C_u\right)^+ \lambda_u^{B,\mathbb{Q}}\, \mathrm{d}u \,\middle|\, \mathcal{F}_t \right],$$

$$
\begin{aligned}
(2.15) \qquad \overline{\mathrm{FVA}}_t := {} & B_t^{\tilde{r}} \mathbb{E}^{\mathbb{Q}}\left[\int_t^T \frac{(r_u^{f,l} - r_u)\left(\widehat{V}_u - \overline{\mathrm{XVA}}_u - C_u\right)^+}{B_u^{\tilde{r}}}\, \mathrm{d}u \,\middle|\, \mathcal{F}_t \right] \\
& - B_t^{\tilde{r}} \mathbb{E}^{\mathbb{Q}}\left[\int_t^T \frac{(r_u^{f,b} - r_u)\left(\widehat{V}_u - \overline{\mathrm{XVA}}_u - C_u\right)^-}{B_u^{\tilde{r}}}\, \mathrm{d}u \,\middle|\, \mathcal{F}_t \right],
\end{aligned}
$$

$$(2.16) \qquad \overline{\mathrm{ColVA}}_t := B_t^{\tilde{r}} \mathbb{E}^{\mathbb{Q}}\left[\int_t^T \frac{(r_u^{c,l} - r_u)C_u^+ - (r_u^{c,b} - r_u)C_u^-}{B_u^{\tilde{r}}}\, \mathrm{d}u \,\middle|\, \mathcal{F}_t \right].$$

This representation highlights that the inclusion of different borrowing and lending rates introduces a non-zero funding adjustment which cannot be found independently of the other adjustments. As

a consequence, differently from CVA, DVA and ColVA, FVA presents a recursive structure[1]. An algorithm to compute all valuations adjustments systematically in the 'non-recursive' and 'recursive' setting, especially with the view of potentially large portfolios, is the focus of the next sections.

3. The algorithm

In this section, we introduce the algorithm for computing valuation adjustments by neural network approximations to the BSDE model from the previous section. We start by briefly recalling the main features of the Deep BSDE Solver in Han et al. (2018). Then, we present the application of the solver to valuation adjustments and its extensions to obtain financially important quantities. We first focus on non-recursive adjustments, namely CVA and DVA, and then extend the approach to the recursive case (see the terminology introduced at the end of the last section).

In particular, we propose to use the Deep BSDE Solver in Han et al. (2018) to approximate the dynamics of $\widehat{V}_u^m$, $m = 1, \ldots, M$, $u \in [t, T]$, which constitute the portfolio $\widehat{V}_u = \sum_{i=1}^{M} \widehat{V}_u^m$. Once the portfolio value has been approximated and resulting collaterals computed, the value of the adjustment can be obtained either by inserting the values in an 'outer' Monte Carlo computation for non-recursive adjustments, or applying a second time the Deep BSDE Solver to (2.10) in the recursive case.

3.1. **The Deep BSDE Solver of Han et al. (2018).** For the reader's convenience, we describe in this section the main principles of the algorithm in Han et al. (2018) as they are relevant to our setting. We consider a general FBSDE framework.

Let $(\Omega, \mathcal{F}, \mathbb{Q})$ be a probability space rich enough to support an $\mathbb{R}^d$-valued Brownian motion $W^{\mathbb{Q}} = (W_t^{\mathbb{Q}})_{t \in [0,T]}$. Let $\mathbb{F} = (\mathcal{F}_t)_{t \in [0,T]}$ be the filtration generated by $W^{\mathbb{Q}}$, assumed to satisfy the standard assumptions. Let us consider an FBSDE in the following general form:

$$
(3.1) \qquad X_t = x + \int_0^t b\left(s, X_s\right) \mathrm{d}s + \int_0^t a\left(s, X_s\right)^\top \mathrm{d}W_s^{\mathbb{Q}}, \quad x \in \mathbb{R}^d
$$

$$
(3.2) \qquad Y_t = \vartheta(X_T) + \int_t^T h\left(s, X_s, Y_s, Z_s\right) \mathrm{d}s - \int_t^T Z_s^\top \mathrm{d}W_s^{\mathbb{Q}}, \quad t \in [0, T],
$$

where the vector fields $b : [0,T] \times \mathbb{R}^d \mapsto \mathbb{R}^d$, $a : [0,T] \times \mathbb{R}^d \mapsto \mathbb{R}^{d \times d}$, $h : [0,T] \times \mathbb{R}^d \times \mathbb{R} \times \mathbb{R}^d \mapsto \mathbb{R}$ and $\vartheta : \mathbb{R}^d \mapsto \mathbb{R}$ satisfy suitable assumptions ensuring existence and uniqueness results. We denote by $(X_t^x)_{t \in [0,T]} \in \mathbb{S}^2(\mathbb{R}^d)$ and $(Y_t^y, Z_t)_{t \in [0,T]} \in \mathbb{S}^2(\mathbb{R}) \times \mathbb{H}^{2,q \times 1}$ the unique adapted solution to (3.1) and (3.2), respectively. To alleviate notations, hereafter we omit the dependency on the initial condition x of the process $X_\cdot^x$.

The above formulation of FBSDEs is intrinsically linked to the following stochastic optimal control problem:

$$
(3.3) \qquad \underset{y,\ Z=(Z_t)_{t \in [0,T]}}{\text{minimise}} \quad \mathbb{E}\left[\left|\vartheta(X_T) - Y_T^{y,Z}\right|^2\right]
$$

$$
(3.4) \qquad \text{subject to} \quad \begin{cases} X_t &= x + \int_0^t b\left(s, X_s\right) \mathrm{d}s + \int_0^t a\left(s, X_s\right)^\top \mathrm{d}W_s^{\mathbb{Q}}, \\ Y_t^{y,Z} &= y - \int_0^t h\left(s, X_s, Y_s^{y,Z}, Z_s\right)\mathrm{d}s + \int_0^t Z_s^\top \mathrm{d}W_s^{\mathbb{Q}}, \quad t \in [0, T]. \end{cases}
$$

In particular, a solution (Y, Z) to (3.2) is a minimiser of the problem (3.3). A discretized version of the optimal control problem (3.3)–(3.4) is the basis of the Deep BSDE Solver.

---

[1]The ColVA can be recursive or non-recursive depending on the specification of the collateralizaton agreement: if the collateral is a function of the clean value, then the ColVA term is non recursive (and this represent the situation typically found in practice) however, if the collateral depends on the whole value of the transaction, then also this term is recursive as the FVA.
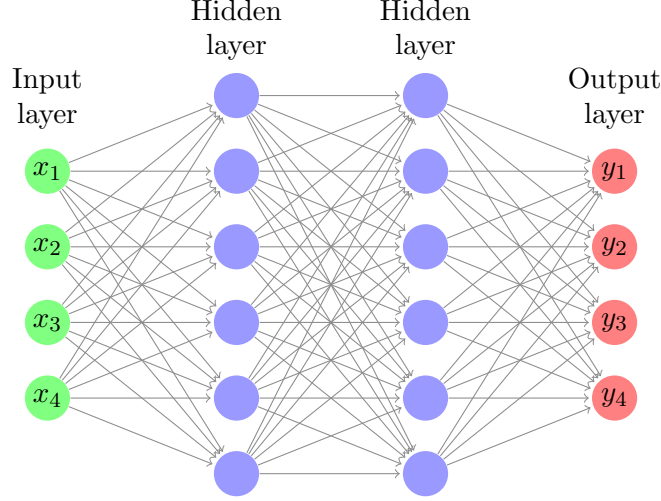
FIGURE 1. Schematic representation of a feedforward neural network with two hidden layers, i.e. $\mathcal{L} = 3$, input and output dimension $d = 4$, and $\nu = d + 2 = 6$ nodes.

Given $N \in \mathbb{N}$, consider $0 = t_0 < t_1 < \ldots < t_N = T$. For simplicity, let us take a uniform mesh with step Δt such that $t_n = n \Delta t$, $n = 0, \ldots, N$, and denote $\Delta W_n = W^{\mathbb{Q}}_{t_{n+1}} - W^{\mathbb{Q}}_{t_n}$. By an Euler-Maruyama approximation of (3.3)–(3.4), one has

$$(3.5) \qquad \widetilde{X}_{n+1} = \widetilde{X}_n + b(t_n, \widetilde{X}_n)\Delta t + a(t_n, \widetilde{X}_n)\Delta W_n, \qquad\qquad \widetilde{X}_0 = x,$$

$$(3.6) \qquad \widetilde{Y}^{y,\widetilde{Z}}_{n+1} = \widetilde{Y}^{y,\widetilde{Z}}_n - h(t_n, \widetilde{X}_n, \widetilde{Y}^{y,\widetilde{Z}}_n, \widetilde{Z}_n)\Delta t + \widetilde{Z}_n^{\top}\Delta W_n, \qquad\qquad \widetilde{Y}^{y,\widetilde{Z}}_0 = y.$$

The core idea of the Deep BSDE Solver is to approximate, at each time step n, the control process $\widetilde{Z}_n$ in (3.6) by using an artificial neural network (ANN). More specifically, in the Markovian setting, Z_t is a measurable function of X_t, which we approximate by an ANN ansatz to carry out the optimisation above over this parametrised form. To this end, we introduce next a formalism for the description of neural networks.

ANN approximation. We consider artificial neural networks with $\mathcal{L} + 1 \in \mathbb{N} \setminus \{1, 2\}$ layers. Each layer consists of ν_ℓ nodes (also called *neurons*), for $\ell = 0, \ldots, \mathcal{L}$. The 0-th layer represents the *input layer*, while the $\mathcal{L}$-th layer is called the *output layer*. The remaining $\mathcal{L} - 1$ layers are *hidden layers*. For simplicity, we set $\nu_\ell = \nu$, $\ell = 1, \ldots, \mathcal{L} - 1$. The input and output dimensions are both d in our case. A feedforward neural network is a function $\varphi^\varrho : \mathbb{R}^d \mapsto \mathbb{R}^d$, defined via the composition

$$x \in \mathbb{R}^d \longmapsto \mathcal{A}_{\mathcal{L}} \circ \varrho \circ \mathcal{A}_{\mathcal{L}-1} \circ \ldots \circ \varrho \circ \mathcal{A}_1(x) \in \mathbb{R}^d,$$

where all $\mathcal{A}_\ell$, $\ell = 1, \ldots, \mathcal{L}$, are affine transformations

$$\mathcal{A}_1 : \mathbb{R}^d \mapsto \mathbb{R}^\nu, \qquad \mathcal{A}_\ell : \mathbb{R}^\nu \mapsto \mathbb{R}^\nu, \quad \ell = 2, \ldots, \mathcal{L} - 1, \qquad \mathcal{A}_{\mathcal{L}} : \mathbb{R}^\nu \mapsto \mathbb{R}^d,$$

of the form $\mathcal{A}_\ell(x) := \mathcal{W}_\ell x + \beta_\ell$, $\ell = 1, \ldots, \mathcal{L}$, where $\mathcal{W}_\ell$ and β_ℓ are matrices and vectors of suitable size called, respectively, weights and biases. The function ϱ, called *activation function* is a univariate function $\varrho : \mathbb{R} \mapsto \mathbb{R}$ that is applied component-wise to vectors. With an abuse of notation, we denote $\varrho(x_1, \ldots, x_\nu) = (\varrho(x_1), \ldots, \varrho(x_\nu))$. The elements of $\mathcal{W}_\ell$ and β_ℓ are the parameters of the neural network. We can regroup all parameters in a vector $\rho \in \mathbb{R}^R$ where $R = \sum_{\ell=0}^{\mathcal{L}} \nu_\ell(1 + \nu_\ell)$.

As indicated earlier, we use ANNs to approximate the control process Z_t. More specifically, let $R \in \mathbb{N}$ as before and let $\xi \in \mathbb{R}$, $\rho \equiv (\rho_1 \ldots, \rho_R) \in \mathbb{R}^R$ be $R + 1$ parameters. We introduce a family of neural networks $\varphi_n^\rho : \mathbb{R}^d \to \mathbb{R}^d$, $n \in \{0, \ldots, N\}$ parametrized by ρ and indexed by time. We denote

$\mathcal{Z}_n^\rho = \varphi_n^\rho(\widetilde{X}_n)$ and consider the following parametrized version of (3.6)

$$(3.7) \qquad \mathcal{Y}_{n+1}^{\xi,\rho} = \mathcal{Y}_n^{\xi,\rho} - h(t_n, X_n, \mathcal{Y}_n^{\xi,\rho}, \mathcal{Z}_n^\rho)\Delta t + (\mathcal{Z}_n^\rho)^\top \Delta W_n, \qquad\qquad \mathcal{Y}_0^{\xi,\rho} = \xi,$$

meaning that, at each time step, we use a distinct neural network to approximate the control process. The Deep BSDE Solver by Han et al. (2018) considers the following stochastic optimization problem

$$(3.8) \qquad \underset{\xi\in\mathbb{R},\, \rho\in\mathbb{R}^R}{\text{minimise}} \ \mathbb{E}\left[\left(\vartheta(\widetilde{X}_N) - \mathcal{Y}_N^{\xi,\rho}\right)^2\right] \quad \text{subject to (3.5)–(3.7).}$$

Observe that, in practice, one simulates $L \in \mathbb{N}$ Monte Carlo paths $(\widetilde{X}_n^{(\ell)}, \mathcal{Y}_n^{\xi,\rho,(\ell)})_{n=0\ldots N}$ for $\ell = 1, \ldots, L$, using (3.5)–(3.7) with N i.i.d. Gaussian random variables $(\Delta W_n)_{n=0,\ldots,N-1}$ with mean 0 and variance Δt. Replacing the expected cost functional by the empirical mean, (3.8) becomes

$$(3.9) \qquad \underset{\xi\in\mathbb{R},\, \rho\in\mathbb{R}^R}{\text{minimise}} \ \frac{1}{L}\sum_{\ell=1}^{L}\left(\vartheta(\widetilde{X}_N^{(\ell)}) - \mathcal{Y}_N^{\xi,\rho,(\ell)}\right)^2 \quad \text{subject to (3.5)–(3.7).}$$

This minimization typically involves a huge number of parameters and it is performed by a stochastic gradient descent-type algorithm (SGD), leading to random approximations. For further details on this point we refer the reader to Section 2.6 in Han et al. (2018). We will denote by $\mathcal{I}$ the maximum number of SGD iterations. To improve the performance and stability of the ANN approximation, a *batch normalization* is also considered, see Ioffe and Szegedy (2015).

The accuracy of the solution is determined by the number of timesteps, number of samples, the chosen network architecture, and the quality of the optimiser found by the chosen optimisation routine. Our practical experience shows that quantifying and controlling the errors resulting from the latter two contributions is particularly difficult. Therefore, certain *a posteriori* error bounds as found in Bender and Steiner (2013) for decoupled FBSDEs, Han and Long (2020) for partially coupled FBSDEs, and in Reisinger et al. (2020) for fully coupled BSDEs are particularly valuable. Specifically, in Han and Long (2020, Theorem 1') the authors show that under suitable assumptions on the coefficients of the FBSDE (3.1)–(3.2), namely, in the decoupled case (see their Assumption 3, 2.), the uniform Lipschitz continuity in space, uniform 1/2-Hölder continuity in time of b, a, h and the Lipschitz continuity of ϑ) one has, for Δt sufficiently small,

$$(3.10) \qquad \sup_{t\in[0,T]} \mathbb{E}|Y_t - Y_t^{\xi,\rho}|^2 + \int_0^T \mathbb{E}|Z_t - Z_t^\rho|^2\,\mathrm{d}t \leq C\Big(\Delta t + \mathbb{E}\Big[\left(\vartheta(\widetilde{X}_N) - \mathcal{Y}_N^{\xi,\rho}\right)^2\Big]\Big),$$

where C is a constant independent of Δt and d possibly depending on the starting point x of the forward process and, given $(\mathcal{Y}_n^{\xi,\rho}, \mathcal{Z}_n^\rho)_{n=0,\ldots,N}$ from (3.7), $Y_t^{\xi,\rho} = \mathcal{Y}_n^{\xi,\rho}$ and $Z_t^\rho = \mathcal{Z}_n^\rho$ for $t \in [t_n, t_{n+1})$.

In Han and Long (2020, Theorem 2'), *a priori* estimates on the term $\mathbb{E}[(\vartheta(\widetilde{X}_N) - \mathcal{Y}_N^{\xi,\rho})^2]$ appearing in the right hand side of (3.10) are also provided. However, the obtained bounds depend on the (unknown) approximation capacity of the considered ANN.

In addition to the combined error bound on the Y and Z approximations in (3.10), we can bound the error for Y in terms of the error for Z, as shown in Appendix A:

$$(3.11) \qquad \sup_{t\in[0,T]} \mathbb{E}|Y_t - Y_t^{\xi,\rho}|^2 \leq C\left(\mathbb{E}|Y_0 - \xi|^2 + \int_0^T \mathbb{E}|Z_t - Z_t^\rho|^2\,\mathrm{d}t\right),$$

for a constant C that only depends on the model parameters (but not ξ or ρ). The controls of the Deep BSDE Solver directly influence the terms on the right-hand side by choice of ξ and ρ. The numerical tests in Section 4 indicate that in our applications ξ can typically be determined more easily and accurately than ρ, and that the errors in Y and Z are of similar magnitude.

Remark 3.1. Let us comment on our use of the the Deep BSDE Solver.

- The new and distinctive feature of our approach consists in employing the deep solver to perform scenario simulations, i.e. simulations of the evolution of the mark-to-market of a portfolio of claims. This shows, for the first time, that the deep solver can be successfully used to solve risk-management problems such as the calculation of xVAs and risk measures (e.g. Value-at-Risk, Expected Shortfall).
- Our use of the Deep BSDE Solver provides an alternative to nested Monte Carlo simulations and their competitors such as regression Monte Carlo (see Longstaff and Schwartz (2001)). Indeed, for the calculation of xVAs we are not only interested in the initial value of the BSDE solution (which is the optimal ξ given as output by the Deep BSDE Solver), but we need the ability to simulate the evolution of the conditional expectations representing the mark-to-market of the portfolio.
- We also observe that our algorithm is interesting in comparison to, e.g., the commonly used regression approaches because we do not only solve for the evolution of the value function (i.e. the portfolio value), but we also obtain the control (i.e., the hedging strategy), which is linked to the sensitivies of the value function. This is of paramount importance in order to calculate many risk measures such as initial margin according to the ISDA Simm methodology. This feature means that our use of the Deep BSDE Solver is useful also in a low dimensional setting, where traditional numerical techniques suffering from the curse of dimension are still viable.

3.2. **The Deep xVA Solver for non-recursive valuation adjustments.** In our setting, the Deep BSDE Solver is first employed in the approximation of the clean values of the portfolio, i.e., the processes $\widehat{V}_t^m$ for $m = 1, \ldots, M$, which are the solutions of (2.6) with underlying forward dynamics given by S in (2.1). More precisely, in the notation of the previous section, we take

$$X_t = S_t \quad \text{and} \quad Y_t = \widehat{V}_t^m \quad \text{for } m = 1, \ldots, M.$$

For simplicity, let us assume $T_m = T$, $\forall m = 1, \ldots, M$. We now describe the algorithm for computing CVA and DVA given by formulas (2.13) and (2.14), respectively. A unifying formula for CVA and DVA can be written as

$$(3.12) \qquad \mathbb{E}^{\mathbb{Q}} \left[\int_t^T \Phi_u(\widehat{V}_u) \, \mathrm{d}u \, \middle| \, \mathcal{F}_t \right],$$

where

- $\Phi_u(v) = (1 - R^C) \frac{B_t^{\tilde{r}}}{B_u^{\tilde{r}}} (v - C(v))^- \lambda_u^{C,\mathbb{Q}}$ for CVA;
- $\Phi_u(v) = (1 - R^B) \frac{B_t^{\tilde{r}}}{B_u^{\tilde{r}}} (v - C(v))^+ \lambda_u^{B,\mathbb{Q}}$ for DVA.

Here, $\Phi_u(v)$ indicates that Φ is a random field. One can easily observe that, thanks to the boundedness of the processes $\tilde{r}$ and λ^j, $j \in \{B, C\}$, $\Phi_u(v)$ is uniformly Lipschitz continuous in v. We denote by L_Φ its Lipschitz constant.

Given a time discretization (uniform, for simplicity) with time step Δt, the integral in (3.12) can be approximated by a quadrature rule, i.e., taking $t = t_0 = 0$,

$$\int_0^T \Phi_u(\widehat{V}_u) \, \mathrm{d}u \approx \sum_{n=0}^N \eta_n \Phi_{t_n}(\widehat{V}_{t_n}).$$

For instance, one may consider the rectangle rule, i.e. $\eta_N = 0, \eta_n = \Delta t \, n = 0, \ldots, N-1$,

$$(3.13) \qquad \int_0^T \Phi_u(\widehat{V}_t) \, dt \approx \sum_{n=0}^{N-1} \Phi_{t_n}(\widehat{V}_{t_n}) \Delta t.$$

Denoting for any $m = 1, \ldots, M$ by $\left(\widehat{\mathcal{V}}_n^{m,\xi_m^*,\rho_m^*,(p)}\right)_{n=0,\ldots,N,p=1,\ldots,P}$ the approximation of P paths of the process $(\widehat{V}_{t_n}^m)_{n=0,\ldots,N}$ obtained by means of the parameters (ξ_m^*, ρ_m^*) resulting from the Deep BSDE Solver optimization (3.9) and

$$\widehat{\mathcal{V}}_n^{*,(p)} := \sum_{m=1}^{M} \widehat{\mathcal{V}}_n^{m,\xi_m^*,\rho_m^*,(p)}, \qquad n = 0, \ldots N,$$

the adjustment is then approximated by the following formula:

$$\frac{1}{P} \sum_{p=1}^{P} \sum_{n=0}^{N} \eta_n \Phi_{t_n}(\widehat{\mathcal{V}}_n^{*,(p)}).$$

Here, P denotes the number of Monte Carlo paths used for estimating the outer expectation in (3.12) which are typically different from the L paths generated for training the NN. Algorithms 1 and 2 summarize the main steps of the method. In what follows we will also denote by $\widehat{V}^{m,\xi_m^*,\rho_m^*}$ the piecewise constant interpolation of $\widehat{\mathcal{V}}^{m,\xi_m^*,\rho_m^*}$.

Algorithm 1: Deep algorithm for exposure simulation

Set parameters: N, L, B.   ▷ *N time steps, L paths for inner Monte Carlo loop, B batch size*
Fix architecture of ANN.   ▷ *intrinsically defines the number of parameters R*
Deep BSDE Solver *(N,L, B)*:

  Simulate L paths $(\widetilde{S}_n^{(\ell)})_{n=0,\ldots,N}$, $\ell = 1, \ldots, L$ of the forward dynamics.
  Define the neural networks $(\varphi_n^\rho)_{n=1,\ldots,N}$.
  for $m = 1, \ldots, M$ **do**
  | Minimize over ξ and ρ
  |
  | $$\frac{1}{L} \sum_{\ell=1}^{L} \left(g_m(\widetilde{S}_N^{(\ell)}) - \widehat{\mathcal{V}}_N^{m,\xi,\rho,(\ell)} \right)^2,$$
  |
  | subject to
  |
  | (3.14) $$\begin{cases} \widehat{\mathcal{V}}_{n+1}^{m,\xi,\rho,(\ell)} = \widehat{\mathcal{V}}_n^{m,\xi,\rho,(\ell)} + r_{t_n} \widehat{\mathcal{V}}_n^{m,\xi,\rho,(\ell)} \Delta t + (\widehat{\mathcal{Z}}_n^{m,\rho,(\ell)})^\top \Delta W_n^{(\ell)}, \\ \widehat{\mathcal{V}}_0^{m,\xi,\rho,(\ell)} = \xi, \\ \quad \widehat{\mathcal{Z}}_n^{\rho,(\ell)} = \varphi_n^\rho(\widetilde{S}_n^{(\ell)}). \end{cases}$$
  |
  | Save the optimizer (ξ_m^*, ρ_m^*).
  end
end

Under reasonable assumptions, we can derive the following *a posteriori* bounds for the error associated with this approximation of the valuation adjustments in $[0, T]$, starting from (3.10). The derivation is given in Appendix B. We note that these adjustments can also be obtained from the more general framework in Section 3.3, however, we provide a simpler numerical procedure here and derive error estimates for these approximations by a more explicit computation.

Algorithm 2: Deep xVA Solver for non-recursive valuation adjustments

---

Apply Algorithm 1

Set parameters: P.   ▷ *P paths for the outer Monte Carlo loop*

Simulate, for $m = 1 \ldots M$, $\big(\widehat{\mathcal{V}}_n^{m,\xi_m^*,\rho_m^*,(p)}\big)_{n=0\ldots N, p=1\ldots P}$ by means of (3.14) with $(\xi,\rho) = (\xi_m^*, \rho_m^*)$.

▷ *approximation of the clean values*

Define $\widehat{\mathcal{V}}_n^{*,(p)} := \sum_{m=1}^M \widehat{\mathcal{V}}_n^{m,\xi_m^*,\rho_m^*,(p)}$ for $n = 0 \ldots N$, $p = 1 \ldots P$.

▷ *approximation of the clean portfolio value*

Compute the adjustment as

$$\frac{1}{P} \sum_{i=1}^P \left(\sum_{n=0}^N \eta_n \Phi_{t_n}(\widehat{\mathcal{V}}_n^{*,(p)}) \right).$$

Let $\widehat{V}_t = \sum_{m=1}^M \widehat{V}_t^m$ with $\widehat{V}_t^m$ given by (2.6), and $\widehat{\mathcal{V}}_n^* = \sum_{m=1}^M \widehat{\mathcal{V}}_n^{m,\xi_m^*,\rho_m^*}$ $(n = 0, \ldots, N)$ its approximation from the Deep BSDE Solver. Consider the running assumptions of this paper together with uniform Hölder continuity in t of b and σ, and assume estimate (3.10) for equation (2.6).[2]

Moreover, consider the specific forms of Φ above, assuming $\mathbb{E}[(\lambda_s^{j,\mathbb{Q}} - \lambda_t^{j,\mathbb{Q}})^2] \le C(t-s)$ for $0 \le s \le t \le T$ and $j \in \{B, C\}$.[3]

Then, for Φ as above, there exists a constant $C \ge 0$ depending only on the model inputs and the constants coming from (3.10) (in particular not on Δt and the ANN parameters), such that

$$(3.15) \qquad \left| \mathbb{E}\Big[\int_0^T \Phi_t(\widehat{V}_t)\, \mathrm{d}t \Big] - \mathbb{E}\Big[\sum_{n=0}^{N-1} \Delta t \Phi_{t_n}(\widehat{\mathcal{V}}_n^*) \Big] \right| \le C\Big(\Delta t + \sum_{m=1}^M \mathbb{E}\Big[|g_m(S_T) - \widehat{\mathcal{V}}_N^{m,\xi_m^*,\rho_m^*}|^2 \Big] \Big)^{1/2}.$$

MC standard errors for the second expectation in (3.15) should be added to obtain a complete bound.

3.3. **The Deep xVA Solver for recursive valuation adjustments.** The procedure of the previous section is sufficient to perform the estimation of CVA and DVA according to (2.13) and (2.14) at time zero by means of a standard Monte Carlo estimator, given the pathwise solutions of the BSDEs for clean values. Typically, however, the bank needs to also compute risk measures on the CVA, such as Value–at–Risk. Moreover, if we consider the xVA BSDE (2.10), we observe that FVA terms introduce a recursive structure through the driver, so that a time t estimate of the process $\overline{\mathrm{XVA}}$ requires the use of a numerical solver for a BSDE. Finally, let us observe that the bank is not only interested in computing the xVA at time t, also hedging the market risk of xVA is important, meaning that one also needs sensitivities of valuation adjustments with respect to the driving risk factors.

All above considerations motivate us to propose a two-step procedure, where we first employ the Deep BSDE Solver to estimate the clean values $\widehat{V}^m$, $m = 1, \ldots, M$, according to Algorithm 1 and then, using the simulated paths of the M clean BSDEs obtained from the first step, we apply again the Deep BSDE Solver to numerically solve the xVA BSDE (2.10). The procedure is outlined in Algorithm 3. Similar to Section 3.2, we can quantify the error of the Deep xVA Solver in the recursive case *a posteriori*. Let $(\overline{\mathrm{XVA}}_t, \overline{Z}_t)$ be the solution of (2.10), $(\widetilde{\mathrm{XVA}}_t^{\gamma,\zeta}, \widetilde{Z}_t^\zeta)$ the corresponding approximation

---

[2] This is a straightforward extension of Han and Long (2020, Theorem 1') in the case of deterministic r_s in (2.6) which is Hölder-1/2 in s, by replacing their assumption on the uniform Hölder-1/2 continuity of f by $|f(t,v) - f(s,v)| \le C|t - s|^{1/2}|v|$ for all $0 \le s \le t \le T$ and all v. For stochastic rates, a more substantial extension to their analysis is needed for a direct application of Euler-Maruyama, due to the non-Lipschitz term $r_t\widehat{V}_t$ in (2.6) and accounting for the discretisation of the rates process. However, the simple transformation (B.2) from the appendix can eliminate this drift. We hence directly assume (B.1) for this analysis.

[3] This is immediate for deterministic Hölder-1/2 functions and a standard property of Itô diffusions with Lipschitz coefficients (see Zhang (2004, Lemma 2.4, (2.10))), but also holds, e.g., for the Cox-Ingersoll-Ross process (as follows e.g. from Hutzenthaler et al. (2014, Corollary 2.14)).

Algorithm 3: Deep xVA Solver

Apply Algorithm 1.

Set parameters: P. $\triangleright$ *P paths for outer Monte Carlo loop*

Fix architecture of ANN.

$\triangleright$ *intrinsically defines the number of parameters $\bar{R}$ (in general $\bar{R} \neq R$)*

Deep XVA-BSDE solver *(N,P)*:

Simulate P paths $(\mathcal{V}_n^{(p)})_{n=0,\ldots,N}$, $p = 1, \ldots, P$, of the portfolio value.

Define the neural networks $(\psi_n^\zeta)_{n=1,\ldots,N}$.

Minimize over γ and ζ

$$\frac{1}{P} \sum_{p=1}^{P} \left(\overline{\mathcal{X}}_N^{\gamma,\zeta,(p)} \right)^2,$$

subject to

(3.16)
$$\begin{cases} \overline{\mathcal{X}}_{n+1}^{\gamma,\zeta,(p)} = \overline{\mathcal{X}}_n^{\gamma,\zeta,(p)} - \bar{f}(t_n, \widehat{\mathcal{V}}_n^{(p)}, \overline{\mathcal{X}}_n^{\gamma,\zeta,(p)})\Delta t + (\overline{\mathcal{Z}}_n^{\zeta,(p)})^\top \Delta W_n^{(p)}, \\ \overline{\mathcal{X}}_0^{\gamma,\zeta,(p)} = \gamma, \\ \overline{\mathcal{Z}}_n^{\zeta,(p)} = \psi_n^\zeta(\widehat{\mathcal{V}}_n^{(p)}). \end{cases}$$

end

---

from the Deep BSDE Solver with parameters γ, ζ, with $\widehat{V}$ in (2.10) replaced by $\widehat{V}^{\xi,\rho}$ given by the solver with parameters ξ, ρ. We note that the result of Han and Long (2020) can be extended to multi-dimensional BSDEs (see the comment at the start of Section 2 there), or that our system is a special case of the fully-coupled McKean–Vlasov FBSDEs analysed in Reisinger et al. (2020) (where the monotonicity condition H.1.(1) imposed there is not needed here in the weakly coupled case).

Take the running assumptions of this paper. Moreover, let for simplicity all rates and intensity processes be bounded, uniformly 1/2-Hölder continuous deterministic functions of time and the functions μ, σ, $\bar{f}$ be uniformly 1/2-Hölder continuous in time. Then there exists a constant $K \geq 0$ depending only on the model inputs (in particular not on Δt and the ANN parameters) such that

$$\sup_{t \in [0,T]} \mathbb{E}\left[\left| \overline{\mathrm{XVA}}_t - \widetilde{\mathrm{XVA}}_t^{\gamma,\zeta} \right|^2 \right] + \mathbb{E}\left[\int_0^T |\overline{Z}_t - \widetilde{Z}_t^\zeta|^2 \, \mathrm{d}t \right]$$

$$\leq K \left(\Delta t + \sum_{m=1}^{M} \mathbb{E}\left[|g_m(S_T) - \widehat{\mathcal{V}}_N^{m,\xi_m,\rho_m}|^2 \right] + \mathbb{E}\left[|\widetilde{\mathrm{XVA}}_T^{\gamma,\zeta}|^2 \right] \right).$$

It should be possible to derive similar results for bounded or even unbounded stochastic rates, but care would have to be taken with the discretisation in the case of non-Lipschitz coefficients, such as the CIR model.

3.4. **Calculation of risk measures.** An important benefit of the deep xVA solver is given by the ability to compute risk measures as a by-product without additional numerical burden and to do so for any time horizon within the simulation time grid. More specifically, let $\mathcal{P}$ denote a process of interest, which could represent either the clean value $\widehat{\mathcal{V}}^{*,(p)}$ as estimated via Algorithm 1 or the xVA correction $\overline{\mathcal{X}}^{*,(p)}$ as produced by Algorithm 3. Given time points t_n, $n = 1, \ldots, N$, we can define the loss process

$$L_{t_n} := -\left(\mathcal{P}_{t_n} - \mathcal{P}_{t_0} \right).$$

The above defined discrete time stochastic process L can then be used to compute risk measures at each point in time over the simulation grid. To provide examples, we can compute e.g.

- Value at Risk:

$$\mathrm{VaR}_\alpha(\mathcal{P}_{t_n}) := \inf\left\{ l \in \mathbb{R} \middle| \mathbb{Q}\left(L_{t_n} > l \middle| \mathcal{F}_{t_0}\right) \leq 1 - \alpha\right\},$$

- Expected Shortfall:

$$ES_\alpha(\mathcal{P}_{t_n}) := \mathbb{E}\left[L_{t_n} \middle| L_{t_n} \geq \mathrm{VaR}_\alpha(L_{t_n}), \mathcal{F}_{t_0}\right],$$

both on the clean value and, more importantly, on the xVAs. Notice that the computation of risk measures, e.g. on the CVA, does not require the use of nested simulations. We simply simulate the trajectories of the BSDE satisfied by the value adjustment and evaluate the risk measure over the simulated paths. This is demonstrated in Section 4.3, where we compute the Value at Risk on the CVA of a 100-dimensional basket option.

3.5. **Pathwise simulation of sensitivities.** One interesting feature of our approach to xVA computations is that we can easily estimate several sensitivities (i.e., partial derivatives) of pricing functions. Let us recall that, in the present Markovian setting, the control Z associated with a FBSDE of the general form (3.1)–(3.2) satisfies

$$(3.17) \qquad Z_t = \frac{\partial Y}{\partial X}(t, X_t) a(t, X_t),$$

so that we can easily reconstruct the gradient of the pricing function with respect to all risk factors simply by multiplying each (vector-valued) neural network by the inverse (assuming it exists) of the matrix $a(t, X_t)$. This becomes particularly interesting in view of Algorithms 1 and 3, where we can obtain hedge ratios both for the clean value and for the valuation adjustments without further computations.

Obtaining second order sensitivities, which may also be important for hedging purposes, is also feasible in our setting, because feedforward neural networks are compositions of simple functions and computation of gradients of neural network functions has become standard in that community. Using the notation of Section 3.1, we can write

$$(3.18) \qquad \frac{\partial \mathcal{Z}_n^\rho}{\partial X_n} = \frac{\partial \varphi^\varrho(X_n)}{\partial X_n},$$

with $\varphi^\varrho(X_n) = \mathcal{A}_\mathcal{L}(\rho(\mathcal{A}_{\mathcal{L}-1} \ldots \rho(\mathcal{A}_1(X_n))))$. Since $(\mathcal{A}_\ell)_{\ell=1,\ldots,\mathcal{L}}$ are affine functions, their Jacobians are given by the weight matrices, i.e.

$$J_{\mathcal{A}_\ell}(\cdot) = \mathcal{W}_\ell, \qquad \ell = 1, \ldots, \mathcal{L}.$$

Moreover, one also has the Jacobian of ρ,

$$J_\varrho(\cdot) = \mathrm{diag}\left(\varrho'(\cdot)\right),$$

where, for $x \in \mathbb{R}^\nu$ we denote $\varrho'(x) = (\varrho'(x_1), \ldots, \varrho'(x_\nu))$. In the present paper, we choose $\varrho(x) = \mathrm{ReLU}(x) = \max\{x, 0\}$ so that the first derivative can be defined as

$$\varrho'(x) = \mathrm{ReLU}'(x) = \left\{ \begin{array}{ll} 1 & \text{if } x > 0 \\ 0 & \text{otherwise} \end{array} \right\} = \mathrm{sgn}(\mathrm{ReLU}(x)).$$

Finally, we deduce that the following explicit differentiation formula holds:

$$\frac{\partial \mathcal{Z}_n^\rho}{\partial X_n} = \mathcal{W}_{\mathcal{L}} \ \text{diag}\left(\varrho'(\mathcal{A}_{\mathcal{L}-1}(\dots \mathcal{A}_1(X_n)))\right) \ \dots \ \text{diag}\left(\varrho'(\mathcal{A}_1(X_n))\right) \mathcal{W}_1.$$

Given the availability of the derivative of $\mathcal{Z}_n^\rho$ we can then obtain the Hessian of Y from (3.17).

4. NUMERICAL RESULTS

To test our algorithm, we start by studying two very simple examples with a similar computational structure as CVA and DVA, and for which we can easily provide reference solutions. We will then give a higher-dimensional example and illustrate further practically relevant features of the method, such as recursive xVA computations and simulation of the collateral account. The codes for the proposed experiments are available at `https://github.com/AlessandroGnoatto/Deep-xVA-Solver`.

Let S be the price of a single stock described by a Black-Scholes dynamics,

$$\mathrm{d}S_t = rS_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t^{\mathbb{Q}}, \qquad S_0 = s_0,$$

and $\widehat{V}$ a European-style contingent claim with value

$$\widehat{V}_t = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-t)}g(S_T)|\mathcal{F}_t\right].$$

In particular, $\widehat{V}$ solves the following BSDE:

$$(4.1) \qquad \begin{cases} -\mathrm{d}\widehat{V}_t = -r\widehat{V}_t\,\mathrm{d}t - \widehat{Z}_t\,\mathrm{d}W_t^{\mathbb{Q}}, \\ \widehat{V}_T = g(S_T). \end{cases}$$

The discounted expected positive and negative exposure of $\widehat{V}$ are defined, respectively, by

$$(4.2) \qquad \mathrm{DEPE}(s) = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(s-t)}\left(\widehat{V}_s\right)^+ \middle| \mathcal{F}_t\right],$$

$$(4.3) \qquad \mathrm{DENE}(s) = -\mathbb{E}^{\mathbb{Q}}\left[e^{-r(s-t)}\left(\widehat{V}_s\right)^- \middle| \mathcal{F}_t\right].$$

In order to take into account the randomness of the algorithm (through the inner and outer Monte Carlo estimation and stochastic gradient descent), in the plots below we report with solid lines the average DEPE (in blue) and DENE (in red) obtained after 100 runs of the algorithm and the gray region represent the obtained standard deviation from the average value.

4.1. **A forward on S.** In this case, we consider

$$g(S_T) = S_T - K$$

with $K = s_0$. The pathwise exposure $\widehat{V}$ at time $s \in [t, T]$ is given by

$$\widehat{V}_s = \mathbb{E}^{\mathbb{Q}}\left[e^{-r(T-s)}(S_T - K)\middle| \mathcal{F}_s\right] = S_s - Ke^{-r(T-s)}.$$

Substituting in (4.2), one has

$$(4.4) \qquad \mathrm{DEPE}(s) = S_t\Phi(d_1) - Ke^{-r(T-t)}\Phi(d_2),$$

$$(4.5) \qquad \mathrm{DENE}(s) = S_t\Phi(-d_1) - Ke^{-r(T-t)}\Phi(-d_2),$$

where $\Phi(\cdot)$ denotes the standard normal cumulative distribution function and, as usual,

$$d_1 = \frac{\ln\left(e^{r(t-s)}S_t/K\right) + \left(r + \sigma^2/2\right)(s-t)}{\sigma\sqrt{s-t}} \quad \text{and} \quad d_2 = d_1 - \sigma\sqrt{s-t}.$$

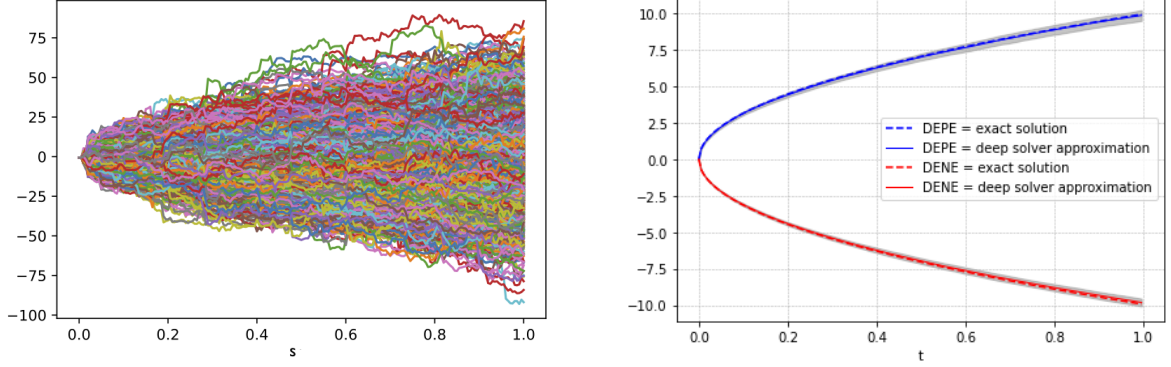| σ	K	T
0.25	100	1

TABLE 1. Parameters used in numerical experiments.

FIGURE 2. Forward contract: approximated exposure (left) and DEPE, DENE (right). Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, $\mathcal{I} = 4000$, time steps $N = 200$.
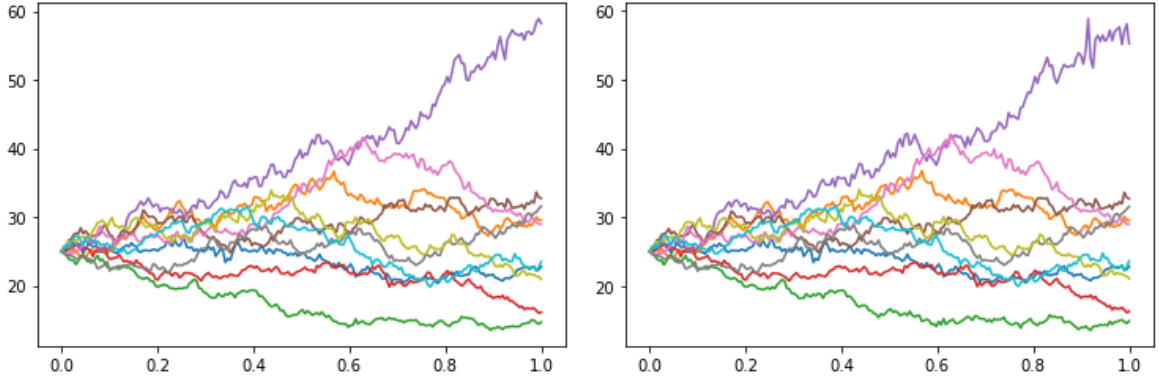

FIGURE 3. The hedging strategy for the forward contract on 10 simulated scenarios: exact (left) and approximated (right).

We report in Figure 2 the plot of the numerical results obtained by Algorithm 2 using the parameters in Table 1 and $r = 0$. In particular, on the left we plot the simulated pathwise exposure, i.e. the paths $t_n \to \widehat{\mathcal{V}}_n^{*,(p)}$ for $p = 1, \ldots, P$ obtained by a single run of the algorithm, while on the right we compare the approximated DEPE and DENE (solid lines) with the exact expected exposures given by (4.4)–(4.5) (dashed lines). The maximum difference between the approximated expected exposure and the exact one is 8.1 bps for the DEPE and 12 bps for the DENE (in both cases achieved at the terminal time T) with a maximal standard deviation of 0.3647.

We asses the performance of the solver in the reconstruction of exposure trajectories computing the average terminal square error on the exposure, i.e. $\frac{1}{P} \sum_{p=1}^{P} (\widehat{\mathcal{V}}_N^{*,(p)} - (S_T^{(p)} - K))^2$ which, for a chosen single run of the algorithm, is found to be 0.1883 (comparable with the loss function given by the solver, which is 0.1664). Moreover, in this special case, we can also compare $\widehat{\mathcal{Z}}^{\rho^*}$ with the exact hedging strategy $\widehat{Z}_t = \sigma S_t$. We display in Figure 3 the exact and approximated hedging strategy, for which we obtain an L^2-norm of the error equal to 0.1759. The CPU time is 503 s.
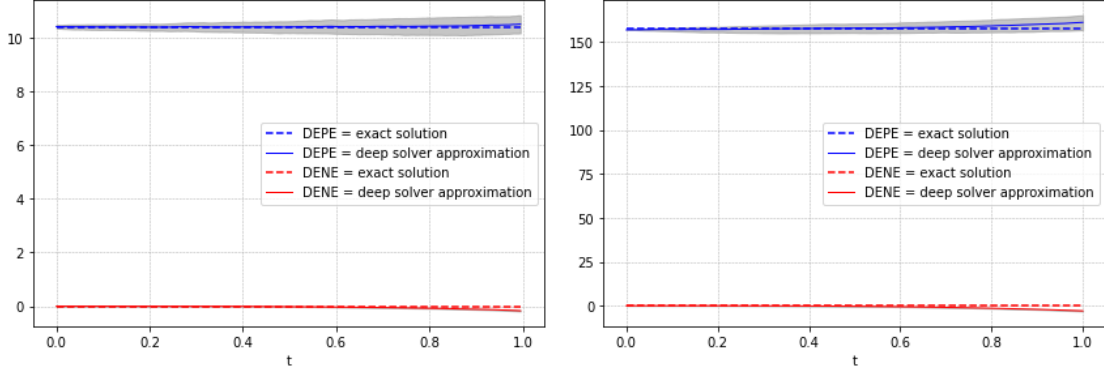
FIGURE 4. DEPE and DENE for a European call option (left) and a European basket option with 100 underlyings (right). Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$ (left) and $\nu = d + 10 = 110$ (right), iterations $\mathcal{I} = 4000$ (left) and $\mathcal{I} = 10000$ (right), time steps $N = 200$ (left) and $N = 100$ (right).

4.2. **A European call option.** In this case we consider

$$g(S_T) = (S_T - K)^+ ,$$

where we set $K = s_0$. The pathwise exposure $\widehat{V}$ at time $s \in [t, T]$ is given by the Black-Scholes formula

$$\widehat{V}_s = \mathbb{E}^{\mathbb{Q}} \left[e^{-r(T-s)} (S_T - K)^+ \middle| \mathcal{F}_s \right] = S_s \Phi(d_1) - K e^{-r(T-s)} \Phi(d_2) > 0.$$

It follows immediately that

$$\mathrm{DEPE}(s) = \mathbb{E}^{\mathbb{Q}} \left[e^{-r(s-t)} \widehat{V}_s \middle| \mathcal{F}_t \right] = \widehat{V}_t,$$

and

$$\mathrm{DENE}(s) = 0.$$

The average terminal square error on the exposure computation, for a chosen run of the algorithm, is given by 0.7894 (comparable with the loss function given by the solver 0.7779). In this case, the exact hedging strategy is $\widehat{Z}_t = \sigma \Phi(d_1) S_t$, from which we can compute the L^2-norm of the error with the approximated control $\widehat{\mathcal{Z}}^{\rho^*}$ which is equal to 0.1496. We report in Figure 5 the exact and the approximated hedging strategy.

The results obtained using Algorithm 2 with the parameters in Table 1 and $r = 0.01$ are reported in Figure 4 (left). The exact European call price is 10.4036, while the approximation of the positive and negative exposure obtained by the solver and reported in Figure 4 (left) take values, for $t \in [0, T]$, within the interval $[10.4072, 10.4963]$ and $[-0.1692, 0]$, respectively. The accuracy of the time zero option value for this architecture and simulation parameters is hence 0.36 bps, and that of DEPE and DENE in the worst case (over s) is 9.3 bps and 17 bps, respectively. The CPU time is 543 s.

We also report in Figure 6 the approximation of the Value at Risk ($\mathrm{VaR}_\alpha(\widehat{V}_t)$, with $\alpha = 0.05$) computed as explained in Section 3.4. Comparing with the exact values of the VaR (dashed line in Figure 6), one can observe a good fit. We point out that for the VaR approximation we imposed the positivity of the value $\widehat{V}$, since without this condition precision was lost close to maturity.

4.3. **A basket call option.** Let us now consider the case of several underlying assets $(S^1, \ldots, S^d)$:

$$dS_t^i = r^i S_t^i \, dt + \sigma^i S_t^i \, dW_t^{\mathbb{Q}, i}, \qquad S_0^i = s_0^i > 0, \qquad i = 1, \ldots, d,$$
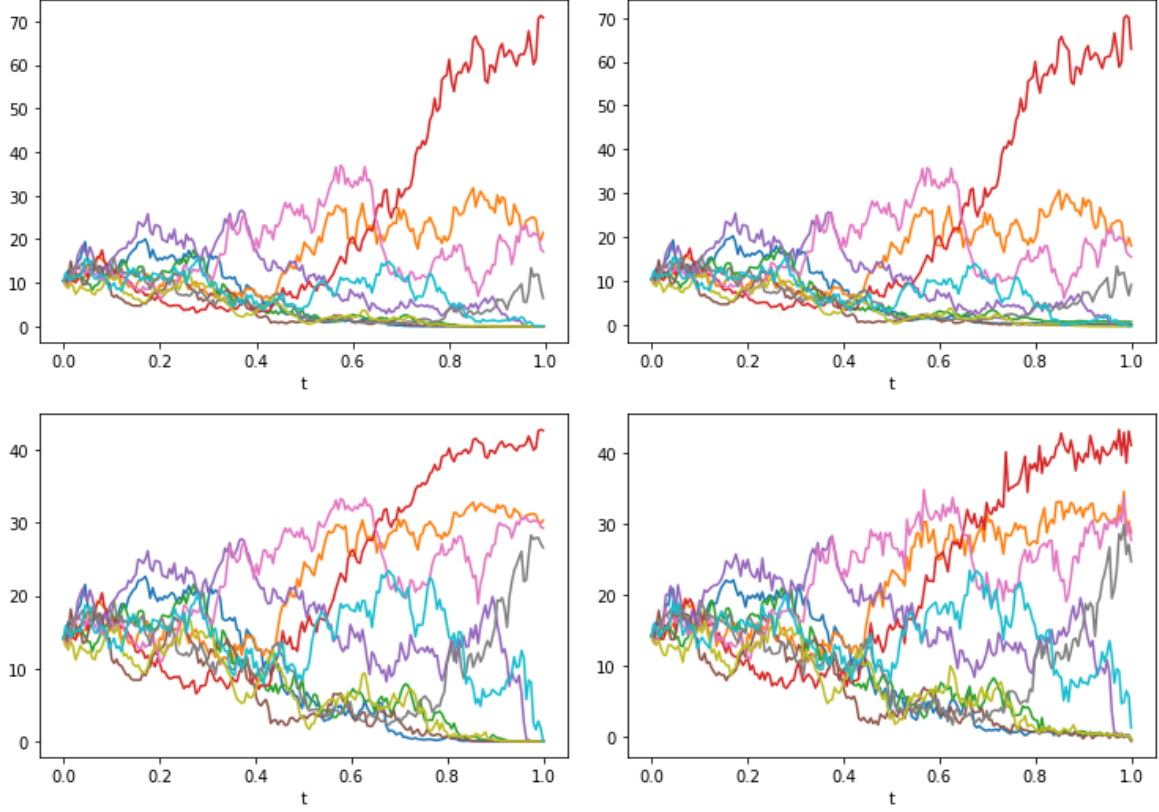
FIGURE 5. The clean value (top row) and hedging strategy (bottom row) for the call option on 10 simulated scenarios: exact (left) and approximated (right).
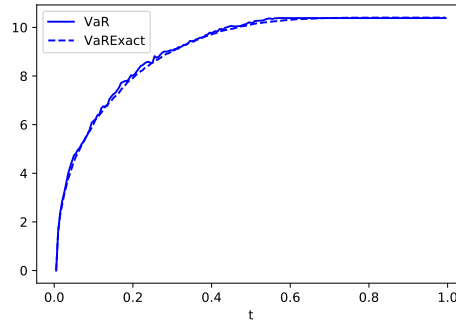

FIGURE 6. Approximation of the $\mathrm{VaR}(\widehat{V}_t)$ for $t \in [0, T]$. Parameters used: see the caption of Figure 4 (left).

where $W^{\mathbb{Q}} = (W^{\mathbb{Q},1}, \ldots, W^{\mathbb{Q},d})$ is a standard Brownian motion in $\mathbb{R}^d$ with correlation matrix $(\rho_{i,j})_{1 \leq i,j \leq d}$. We set $d = 100$. A European basket call option is associated with the payoff

$$g(S_T^1, \ldots, S_T^d) = \left(\sum_{i=1}^{d} S_T^i - d \cdot K \right)^+.$$

The results obtained by Algorithm 2 using the parameters in Table 1 with $\sigma^i = \sigma$ for all $i = 1, \ldots, d$, zero correlation, $s_0^i = 100$ for all $i = 1, \ldots, d$ and $r^i = r = 0.01$ are reported in Figure 4 (right).

The distinctive feature of the present example is the high dimension of the vector of risk factors. While the two previous one-dimensional examples mainly served as a validation for the methodology, the

present example highlights the ability of the proposed methodology to provide an accurate numerical approximation in a high-dimensional context. For this example, we used the feedforward neural network with two layers and $d + 10$ nodes, with a ReLU activation function. The approximation parameters used are reported in the caption of Figure 4 (right). We increase the number of nodes ν roughly linearly with the dimension d, which turned out to be a useful rule-of-thumb for consistent accuracy across dimensions in this case.

For a detailed study of deep learning values of basket derivative (on six underlying asses) from simulated values, not based on BSDEs, see Ferguson and Green (2018).

For the case of the basket call option, we observe that the exposure profile corresponds to the present value of the contract. As a consequence, we obtain a simple method to validate the exposure profile by computing an estimate of the basket call option price by means of a standard Monte Carlo simulation with 10^5 paths. We regard this as the 'exact' price. The Monte Carlo price we obtained is 157.99 with confidence interval $[157.63, 158.34]$. The average values of the expected exposures produced by the deep solver reported in Figure 4 (right) vary with time between the values 156.98 and 161.24 in the positive case, and 0 and -2.9824 in the negative one, achieving at the terminal time $t = T$ the maximum distance 3.25 to the Monte Carlo price in the first case and 2.98 to the exact zero solution in the second case. The accuracy of the time 0 option price is therefore 1bp, and hence of the same order of magnitude as for the single underlying. The CPU time is 1287 s.

Remark 4.1. It is noticeable that the error of DEPE and DENE approximation is relatively low at time zero and eventually increases with time. This is because the time zero value is determined solely by the obtained optimiser for ξ, which is decoupled from the harder optimisation problem for ρ. The optimal ξ which minimises the idealised objective function without time stepping and sampling error is the expected payoff, while ρ determines the ANN hedge which minimises the variance. A suboptimal ANN leads to larger hedging errors, and hence increasing DEPE and DENE, as time increases.

One could use this observation to set ξ to be an accurate MC estimator for the option price, and then minimize over ρ only. This by construction gives accurate time zero values for DEPE and DENE, but from our tests (not reported here) leads to similar results to above for larger t.

In relation to (3.11), this shows that the first term on the right-hand side can be made negligible compared to the second term. In these examples, the error of Y and Z are indeed of similar magnitude. This is supported by (A.1) in the appendix.

For this product, next, we also perform an xVA calculation with the objective to validate Algorithm 2 and Algorithm 3 in a case where both are applicable. To perform this comparison, we need the xVA BSDE to be non-recursive: this can be achieved by assuming that there is a unique risk-free interest rate, so that FVA and ColVA are identically zero, i.e., xVA consists only of the CVA and DVA term. The idea is then to compare a Monte Carlo estimate of xVA according to Algorithm 2 with the initial value of the BSDE as produced by a full application of Algorithm 3.

We assume that the default intensities of the bank and the counterparty are $\lambda^{C,\mathbb{Q}} = 0.10$ and $\lambda^{B,\mathbb{Q}} = 0.01$, respectively. For the recovery rates we set $R^C = 0.3$ and $R^B = 0.4$, while the unique risk-free interest rate is $r = 0.01$. Using the same network setting (see again the caption of Figure 4, right), the Deep xVA Solver produced an xVA estimate of 0.8952 by means of Algorithm 3 (CPU = 3098 s), whereas the estimate produced by Algorithm 2 is 0.8947 with an associated confidence interval $[0.8927, 0.8968]$ (CPU = 1379 s).

As pointed out in Section 3.4, Algorithm 3 can also be used to compute risk measures for xVAs. In Figure 7 (right) we report the plot of the Value at Risk (VaR, with $\alpha = 0.05$) of the xVA, i.e. we
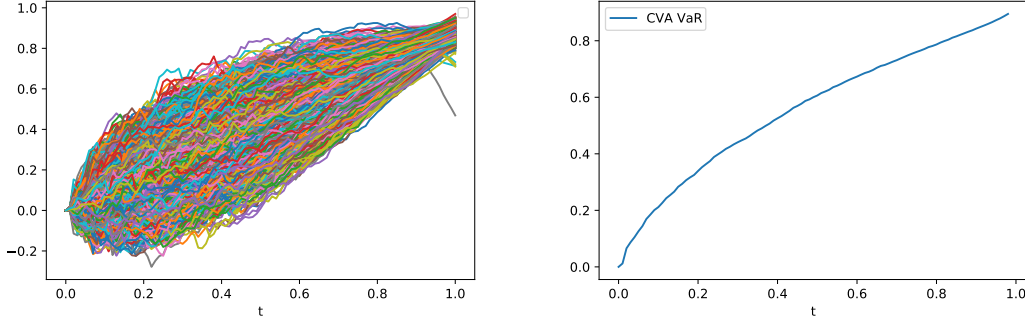
FIGURE 7. Simulated paths of the loss process $L_t = -(\text{XVA}_t - \text{XVA}_0)$ (left) and approximation of the $\text{VaR}(\text{XVA}_t)$ for $t \in [0, T]$ (right).

compute $\text{VaR}_\alpha(\text{XVA}_t)$ taking $L_t = -(\text{XVA}_t - \text{XVA}_0)$, for $t \in [0, T]$, as loss process (simulated paths of the loss are reported in Figure 7 (left)). Considering the same discretization parameters as the test above the required computational time is 3425 s, which confirms that after a single application of Algorithm 3 to solve the xVA BSDE, risk measures can be obtained at a very low cost just by simulating the associated BSDE trajectories. We acknowledge that the computation of risk measures, which focuses on tails of the distribution of the value process, poses some challenges to our method: we faced numerical instabilities due to the representation of floating point numbers in Python that we addressed by exploiting the fact that both the clean value and the xVA are homogenous functions of order one with respect to the notional. We multiply the terminal condition of the clean value BSDE by a scaling factor which we later compensate back after the simulation of the paths of the trained model has been performed. We compute the value at risk for every point in the simulation time grid and we observe a smooth curve that converges towards the terminal value at risk. At time T the loss degenerates to $-(\text{XVA}_T - \text{XVA}_0) = \text{XVA}_0 \sim 0.8952$ due to the fact that the xVA BSDE has a zero terminal condition, hence we have again a test value against which we can compare our estimate given by $\text{VaR}(\text{XVA}_T) = 0.9097$.

4.4. **Recursive FVA computation.** In this section, we provide an FVA calculation that serves as a further validation of Algorithm 3 for recursive valuation adjustments. For the sake of illustration, we simplify the framework of Biagini et al. (2021) so that we recover the funding equations of Piterbarg (2010). More specifically, we assume that there is no default risk, i.e. $\tau^C = \tau^B = +\infty$. We consider the case of a bank trading a forward on a single underlying stock, in line with Example 4.1. We set $r^{c,b} = r^{c,l} = r = 0.02$, and $r^{f,b} = r^{f,l} = 0.04$. Due to the different interest rates for funding and collateral, the clean value of the contract is not at par. For the moment we assume that the claim is perfectly uncollateralized, i.e. $C_t \equiv 0$ $d\mathbb{Q} \otimes dt$-a.s. In this case, as first shown in Piterbarg (2010) and then Biagini et al. (2021) among others, one can employ a risk neutral valuation formula where the discount rate is given by the unsecured funding rate $r^f = r^{f,b} = r^{f,l}$. Precisely, we can write the solution of the pricing problem as

$$V_t = \widehat{V}_t - \overline{\text{FVA}}_t, \qquad \text{where}$$

$$\overline{\text{FVA}}_t = B_t^r \mathbb{E}^{\mathbb{Q}} \left[\int_t^T \frac{(r_u^f - r_u)\left(\widehat{V}_u - \overline{\text{FVA}}_u\right)}{B_u^r} \, du \, \middle| \, \mathcal{F}_t \right].$$

r^f	0.04	0.08	0.12
Solver	0.0395	0.1155	0.1897
Exact	0.0392	0.1153	0.1884

TABLE 2. Numerical solution for the FVA for different levels of the unsecured funding rate r^f. Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, iterations $\mathcal{I} = 4000$, time steps $N = 100$.

The analytic computation of the clean value of the forward contract at time t yields $\widehat{V}_0^{\text{exact}} = 1.9801$. The claim is however uncollateralized, hence, by applying a risk neutral valuation formula where the discounting rate is now r^f, we obtain $V_0^{\text{exact}} = 1.9409$. The difference between the two analytic computations provides us with the exact value of the FVA, i.e. $\overline{\text{FVA}}_0^{\text{exact}} = 0.0392$. For this experiment we apply Algorithm 3 with the following parameters that are the same both for the estimation of the clean value and the FVA: we use $N = 100$, $L = 64$, $P = 2048$ and $\mathcal{I} = 4000$. We use two neural networks for the clean value and the FVA both having 2 hidden layers with $d + 20$ nodes. We then apply Algorithm 3 to the xVA BSDE associated with FVA and obtain an initial value of $\overline{\text{FVA}}_0 = 0.0395$, thus a validation of our proposed numerical procedure. We evaluate the performance of the solver in the reconstruction of exposure trajectories computing the average terminal square error on the FVA, i.e. $\frac{1}{P} \sum_{p=1}^{P} (\overline{\mathcal{X}}_N^{*,(p)})^2$ which, for a chosen single run of the algorithm, is found to be 3.36×10^{-5} (comparable with the loss function 3.11×10^{-5} given by the solver).

To further assess the reliability of the algorithm we test the FVA as a function of the unsecured funding rate r^f: as this rate increases, the funding spread has a higher magnitude, meaning that we expect the FVA to increase. Table 2 provides evidence in this regard.

We also tested the performance of Algorithm 3 with the increasing of the number of risk factors. In Table 3 we report the computational time required by Algorithm 3 for computing the FVA for a forward contract written on a basket of d underlyings, for different value of d. The parameters used in the numerical tests are reported in the caption of the table. We observe that the numerical error is below 1% at least up to dimension 200.

d	Deep XVA Sol.	Exact Sol.	Error	CPU(s)
1	0.03950	0.03920	0.0003	605
10	0.39199	0.39209	0.0001	753
25	0.97568	0.98023	0.0046	803
50	1.9439	1.9605	0.0166	960
100	3.8976	3.9209	0.0233	1410
150	5.8603	5.8813	0.0210	3085
200	7.8159	7.8418	0.0258	3918

TABLE 3. Comparing the computational time and the error for the approximation of the FVA for a forward written on a basket of d underlyings, with $d = 1, 10, 25, 50, 100, 150, 200$. Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20$, iterations $\mathcal{I} = 4000$, time steps $N = 100$.

4.5. **Adding collateral.** A useful feature of our proposed approach consists in the possibility of performing realistic simulations of the collateral account without resorting to simplifying assumptions.

4.5.1. *Realistic simulation of the collateral account.* We can in fact compute the overall outstanding exposure between the bank and the counterparty by the following steps. Algorithm 1 allows us to simulate paths for all processes $\widehat{V}^m$, $m = 1, \ldots, M$. Such paths can then be aggregated so as to produce a simulation of the portfolio process $\widehat{V} = \sum_{m=1}^{M} \widehat{V}^m$, that corresponds to the *pre-collateral exposure*. After this, we compute the value of the collateral balance C corresponding to the simulated paths of $\widehat{V}$, which in turn allows us to compute the *post-collateral exposure* process $\widehat{V} - C$ that enters the xVA formulas.

For illustration, we consider $M = 1$ and the equity forward from the first example. We introduce a simple example of a collateral agreement where collateral is exchanged between the counterparties at every point in time (a margin call frequency that does not coincide with the simulation time discretization can of course be treated as well). Collateral is exchanged only in case the pre-collateral exposure is above (below) a receiving (posting) threshold which are both set equal to 5, i.e.

$$C_t := C(\widehat{V}_t) = (\widehat{V}_t - 5)^+ - (\widehat{V}_t + 5)^-.$$

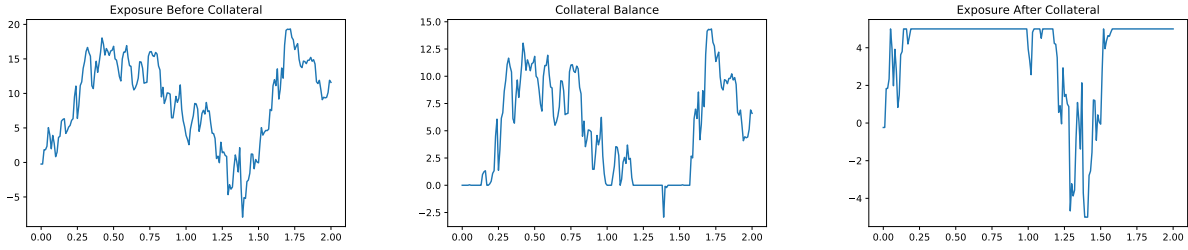An illustration for a single path is provided in Figure 8.

FIGURE 8. Pathwise simulation of a collateralized exposure. Left: $\widehat{V}$. Middle: C. Right: $\widehat{V} - C$. Posting and receiving threshold are 5 EUR.

4.5.2. *Recursive FVA computation in presence of collaterals.* We provide further evidence on the algorithm by studying the impact of collateral more closely. Let us assume that the collateral account is parametrized as follows:

$$C(\widehat{V}) = (1 - \alpha)[(\widehat{V} - 10\alpha)^+ - (\widehat{V} + 10\alpha)^-], \ \alpha \in [0, 1]$$

so that the case $\alpha = 0$ corresponds to perfect collateralization and $\alpha = 1$ to the uncollateralized case. We compute the FVA of a forward on a basket of underlyings, i.e. we consider the terminal condition

$$g\left(S_T^1, \ldots, S_T^d\right) = \left(\sum_{i=1}^{d} S_T^i - d \cdot K\right),$$

for $K = 100$ where the underlying assets evolve as in Section 4.3 and we keep the same choices for the model parameters. We make use of Algorithm 3 assuming $\overline{\text{CVA}} = \overline{\text{DVA}} = \overline{\text{ColVA}} \equiv 0$.

The results are reported in Table 4. We observe, in line with our expectations that in the presence of collateral the FVA becomes numerically negligible. The magnitude of the FVA increases as the level of collateralization decreases. Also we observe that, in the uncollateralized case, the numerical solution provided by the solver closely matches the exact solution for the FVA irrespective of the dimension of the basket of underlying.

d	$\alpha = 0$	$\alpha = 0.5$	$\alpha = 1$	Exact Sol. Uncoll.
1	4.8408×10^{-5}	0.02222	0.03950	0.03920
10	2.8100×10^{-5}	0.21164	0.39199	0.39209
25	-7.8600×10^{-6}	0.48184	0.97568	0.98023
50	1.2700×10^{-5}	0.72473	1.9439	1.9605
100	-6.5900×10^{-6}	0.91248	3.8976	3.9209

TABLE 4. Computation of the FVA for a forward written on a basket of d underlyings. $\alpha = 0$ means full collateralization, where $\alpha = 1.0$ means no collateralization. In the last column we report the exact solution for the no collateral case.

5. COMPARISON WITH OTHER RECENT APPROACHES

In this section, we compare our methodology with other recently developed approaches for portfolio-wide xVA computations from the literature.

5.1. **Gaussian process regression.** In Crépey and Dixon (2020), a Gaussian process regression approach is proposed to perform the computation of derivative portfolio values in the context of (non-recursive) CVA valuation. The Gaussian process approximation of the pricing functional is trained on a set of model implied prices, i.e., for a given set of model parameters, a sample of prices is generated for different combinations of contract characteristics (e.g. strike price and maturity). It is clear that, to generate training data, an efficient numerical scheme is needed; for example, for the Heston model, in their Section 3.1, the COS method of Fang and Oosterlee (2009) is employed to generate training data. Once the regression has been trained, efficient pricing and sensitivities computations are possible even if the portfolio contains a large number of derivatives. The generation of the training data set appears to be a restrictive aspect of using this methodology, especially in the case where individual derivatives are exposed to a large number of risk factors, such as the 100 dimensional basket example we consider. In that case, the generation of gridded price data (the *mark-to-market cube*) over time and all underlying factors is not possible, as remarked in their Section 3.3, and a small number of samples may have to be chosen.[4] A *divide and conquer* approach allows a significant speed-up through GPU or CPU parallelisation in cases where the portfolio is split up into (asset) classes with a restricted, low number of risk factors.

Another limitation emerges when the Gaussian process regression approach is employed to estimate the Value at Risk of the CVA. To estimate such risk measure, one needs to generate a sample of the random variable $\text{CVA}_{t+\Delta t} - \text{CVA}_t$, representing the change in value of the CVA between t and $t + \Delta t$, $\Delta t > 0$. The application of Gaussian process regression results in a nested loop, which results in a quadratic complexity in terms of the number of simulated scenarios: for P simulated paths, the complexity of the CVA VaR computation of Crépey and Dixon (2020) is P^2. Instead, with our Algorithm 3, we directly attack the CVA BSDE and we learn how to simulate trajectories of the CVA, even in high dimension. To estimate the Value at Risk at every possible time horizon within the simulation time grid, we simulate the CVA BSDE with a fixed sample size so that the numerical complexity of our algorithm is lower by one order: we need to simulate only P paths of the clean value and P paths of the CVA BSDE.

5.2. **Different deep learning-based BSDE solvers.** Within our framework, the Deep BSDE Solver of Han et al. (2018) can be directly substituted by other solvers, provided they give approximations to the backward solution and control processes along sample trajectories of the forward process.

---

[4]One could indeed consider using the deep BSDE solver for this scenario generation.

The solver developed by Huré et al. (2020). This is a regression based scheme for PDEs that runs backward in time making use of dynamic programming arguments. The authors study two versions of the algorithm, one where at each point in time two separate neural networks approximate the value function and the control of the BSDE associated to the PDE, and another one where only the value process is approximated by a neural network whereas the control is recovered by automatic differentiation. For each point in time, a neural network is instantiated, and this is common to our methodology. A study of their implementation of the algorithm reveals a problematic point, which is also stated in the text of their paper, that at each point in time, a new Tensorflow session is started and a new computational graph needs to be created, which represents a time consuming operation. Also note that, in order to use their approach for xVA computation, we would need to store each instance of the Tensorflow session on disc so that we can perform the simulation of the exposures after the training is completed. This appears to us as a further bottleneck of the application of this methodology to our setting. A possible advantage that Huré et al. (2020) identify for their approach is that only local-in-time optimisation problems have to be solved, compared to the global optimisation problem of the Deep BSDE Solver. The advantage of good initial values for the optimisation available from previous timesteps appears to compete against the error accumulation from projecting the value function onto neural networks in each timestep.

The approach of Albanese et al. (2021). This framework uses a regression-based algorithm close in spirit to Huré et al. (2020), coupled with Picard iterations for recursive xVA computations and quantile estimation. The authors present a numerical study for the xVA of a swap portfolio with several counterparties and positions exposed to a total of 40 risk factors. As the single contracts that constitute the portfolio are relatively plain vanilla instruments depending on a low number of risk factors, by a divide and conquer approach the computation of single contracts can be split among several GPU/CPU cores. In principle the regression approach can be extended to cases where individual products depend on a high-dimensional vector of risk factors, such as our high dimensional example for the basket option.

A comparison of the performance of all these approaches and variants to different practically relevant situations would be of interest, but goes beyond the scope of this paper.

6. Conclusions and extensions

The proposed xVA algorithm exploits two useful complementary aspects of the Deep BSDE Solver of Han et al. (2018). First, the formulation as an optimisation problem over a parametrisation of the (Markovian) control of the xVA BSDE, which is carried out by SDE discretisation and path sampling, directly gives both the hedge ratios in approximate functional form and model-based derivative prices along the sample paths. This is amenable to the simulation of exposure profiles, the computation of higher-order Greeks by pathwise differentiation, and allows for the computation of funding and margin variation adjustments as well as xVA hedging. A second aspect of the Deep BSDE Solver is the use of neural networks specifically as parametrisation for the Markovian control. A key advantage results from the approximation power of neural networks in high dimensions, which has the potential to make risk management computations on portfolio level feasible. Moreover, the simple functional form allows standard pathwise sensitivity computations.

Our numerical examples provide a proof of concept, but further systematic testing in realistic application settings is needed. An additional difficulty arises from the non-linear, non-convex parametric

form, which, combined with the large number of parameters, leads to challenging optimisation problems. The expression power of the ANN and the practicalities of the learning process, are extremely active research areas and further developments of the proposed Deep xVA Solver will be informed by the rapidly developing understanding of neural networks in a broader sense.

The application of our proposed scheme is not restricted to the chosen xVA framework. For example, one could in principle apply our methodology to the balance-sheet based model computed in Albanese et al. (2021). In this case, the xVA computation involves multiple recursive valuations (illustrated succinctly in Abbas-Turki et al. (2018, Figure 1)), which can be approached by means of multiple applications of the Deep xVA Solver.

We also emphasise that the Deep xVA Solver can be combined with an existing analytics library: the computation of the mark-to-market cube (i.e., the simulation of all possible scenarios for the clean values over different points in time) represents a classical numerical problem to be solved in order to compute traditional risk figures such as Value-at-Risk or Expected Shortfall (this is often referred to as "Monte Carlo full revaluation approach"). Since most products individually depend on a limited number of risk factors, it may be best to use a traditional numerical scheme, such as a finite difference solver, for at least some of the more vanilla products, and then revaluate the products over different Monte Carlo paths by means of a look-up table over the pre-computed numerical solution. This provides an alternative route with respect to our Algorithm 1 for the simulation of the clean values. However, once we aggregate all mark-to-markets, we end up with an object that depends on a high number of risk factors, so for the computation of xVA our proposed methodology provides a useful tool which allows the recursive computation of valuation adjustments, their hedging strategy, and simulation of collateral.

Also, let us stress that our Algorithm 1 returns not only the clean value but also the sensitivies with respect to the forward SDE. The availability of sensitivities is fundamental in order to hedge exposures and also to perform the calculation of initial margin according to the market standard approach (ISDA Simm). In this sense, Algorithm 1 represents a useful alternative to the above mentioned classical approaches in view of the increased demand of advanced analytics by regulators.

Appendix A. Error bounds for Y in terms of Z

We consider the error between the BSDE solution (Y, Z) and its approximation $(Y^{\xi, \rho}, Z^\rho)$ from the Deep BSDE Solver, satisfying

$$
\begin{aligned}
Y_t &= Y_0 - \int_0^t h\left(s, X_s, Y_s, Z_s\right) \, \mathrm{d}s + \int_0^t Z_s^\top \, \mathrm{d}W_s^{\mathbb{Q}}, \quad t \in [0, T], \\
Y_t^{\xi, \rho} &= \xi - \int_0^t h\left(s, X_s, Y_s^{\xi, \rho}, Z_s^\rho\right) \, \mathrm{d}s + \int_0^t (Z_s^\rho)^\top \, \mathrm{d}W_s^{\mathbb{Q}}, \quad t \in [0, T],
\end{aligned}
$$

respectively. Taking the difference and squaring, by elementary inequalities

$$
|Y_t - Y_t^{\xi, \rho}|^2 \leq 3 \left\{ |Y_0 - \xi|^2 + t \int_0^t \left| h\left(s, X_s, Y_s, Z_s\right) - h\left(s, X_s, Y_s^{\xi, \rho}, Z_s^\rho\right) \right|^2 \, \mathrm{d}s \right.
$$
$$
\left. + \left(\int_0^t (Z_s - Z_s^\rho)^\top \, \mathrm{d}W_s^{\mathbb{Q}} \right)^2 \right\}.
$$

Taking expectations, using for the second term the Lipschitz continuity of h in Y and Z, with constants L_Y and L_Z, respectively, and Itô isometry for the last term,

$$
\mathbb{E}|Y_t - Y_t^{\xi, \rho}|^2 \leq 3 \left\{ \mathbb{E}|Y_0 - \xi|^2 + 2t L_Y^2 \int_0^t \mathbb{E} \left| Y_s - Y_s^{\xi, \rho} \right|^2 \, \mathrm{d}s + (2t L_Z^2 + 1) \int_0^t \mathbb{E} |Z_s - Z_s^\rho|^2 \, \mathrm{d}s \right\}.
$$

By Gronwall's inequality,

$$\mathbb{E}|Y_t - Y_t^{\xi,\rho}|^2 \leq \left\{ \mathbb{E}|Y_0 - \xi|^2 + (2tL_Z^2 + 1) \int_0^t \mathbb{E}\,|Z_s - Z_s^\rho|^2\,\mathrm{d}s \right\} 3\exp(6t^2L_Y^2),$$

which proves (3.11) for a C that only depends on T, L_Y and L_Z.

Although the inequality is generally not sharp, the order 2 strong error of Y_t is typically close to the L^2 error (in t and $\mathbb{Q}$) of Z, as seen numerically in the option pricing examples. This is supported by the following simple calculation. We assume here that $Y_0 = \xi$, justified by the observation that Y_0, the option price at time 0, can be accurately found by Monte Carlo estimation. Then it follows from

$$Y_t = Y_0 + r \int_0^t Y_s\,\mathrm{d}s + \int_0^t Z_s\,\mathrm{d}W_s^{\mathbb{Q}},$$

using an integrating factor $\exp(-rs)$ and similar steps to above,

$$(A.1) \qquad \mathbb{E}|Y_t - Y_t^{\xi,\rho}|^2 = \int_0^t \exp(2r(t-s))\,\mathbb{E}|Z_s - Z_s^{\xi,\rho}|^2\,\mathrm{d}s.$$

APPENDIX B. A POSTERIORI ERROR ESTIMATES FOR NON-RECURSIVE ADJUSTMENTS

The estimates provided in Han and Long (2020) can be applied as follows to the adjustment computation in Subsection 3.2, Algorithm 2. We assume the existence of some constant C such that

$$(B.1) \qquad \sup_{t\in[0,T]} \mathbb{E}\Big[|\widehat{V}_t - \widehat{V}_t^{\xi,\rho}|^2\Big] \leq C\left(\Delta t + \sum_{m=1}^M \mathbb{E}\Big[|g_m(S_T) - \widehat{V}_T^{m,\xi_m,\rho_m}|^2\Big] \right),$$

where $\widehat{V}_t^{\xi,\rho}$ is the ANN approximation associated with parameters $\xi = (\xi_1,\ldots,\xi_M)$ and $\rho = (\rho_1,\ldots,\rho_M)$ (and extended to $[0,T]$ by piecewise constant interpolation) of the clean portfolio value $\widehat{V}_t$.

Under the assumed conditions on Φ (uniformly Lipschitz with constant L_Φ), one directly obtains the following estimates

$$\left| \mathbb{E}\Big[\int_0^T \Phi_t(\widehat{V}_t)\,\mathrm{d}t \Big] - \mathbb{E}\Big[\sum_{n=0}^N \eta_n \Phi_{t_n}(\widehat{V}_{t_n}^{\xi,\rho}) \Big] \right|$$

$$\leq \left| \mathbb{E}\Big[\int_0^T \Phi_t(\widehat{V}_t)\,\mathrm{d}t - \sum_{n=0}^N \eta_n \Phi_{t_n}(\widehat{V}_{t_n}) \Big] \right| + \left| \mathbb{E}\Big[\sum_{n=0}^N \eta_n \Big(\Phi_t(\widehat{V}_{t_n}) - \Phi_{t_n}(\widehat{V}_{t_n}^{\xi,\rho}) \Big) \Big] \right|$$

$$\leq \left| \int_0^T \mathbb{E}\big[\Phi_t(\widehat{V}_t)\big]\,\mathrm{d}t - \sum_{n=0}^N \eta_n \mathbb{E}\big[\Phi_{t_n}(\widehat{V}_{t_n})\big] \right| + \left| \sum_{n=0}^N \eta_n \mathbb{E}\Big[\Big(\Phi_t(\widehat{V}_{t_n}) - \Phi_{t_n}(\widehat{V}_{t_n}^{\xi,\rho}) \Big) \Big] \right|$$

$$\leq Q(\Delta t) + \Big(\sum_{n=0}^N |\eta_n|^2 \Big)^{1/2} \Big(\sum_{n=0}^N \Big| \mathbb{E}\Big[\Phi_{t_n}(\widehat{V}_{t_n}) - \Phi_{t_n}(\widehat{V}_{t_n}^{\xi,\rho}) \Big] \Big|^2 \Big)^{1/2}$$

$$\leq Q(\Delta t) + L_\Phi \Big(\sum_{n=0}^N |\eta_n|^2 \Big)^{1/2} \Big(\sum_{n=0}^N \mathbb{E}\Big[|\widehat{V}_{t_n} - \widehat{V}_{t_n}^{\xi,\rho}|^2 \Big] \Big)^{1/2},$$

where $Q(\Delta t)$ is the error associated with the quadrature rule for the function $\varphi(t) := \mathbb{E}[\Phi_t(\widehat{V}_t)]$. The function φ can be proven to be 1/2-Hölder continuous. Indeed, for $\Phi_t(\widehat{V}_t) = (B_t^{\tilde{r}})^{-1}\Psi(\widehat{V}_t)\lambda_t^{C,\mathbb{Q}}$ (the CVA case, and similar for DVA), denoting $\Psi(\widehat{V}_t) = (1-R^C)(\widehat{V}_t - C(\widehat{V}_t))^-$ Lipschitz in $\widehat{V}_t$,

$$|\phi(t) - \phi(s)| \leq$$
$$\mathbb{E}\Big[\Big| (B_t^{\tilde{r}})^{-1}\Psi(\widehat{V}_t)\Big(\lambda_t^{C,\mathbb{Q}} - \lambda_s^{C,\mathbb{Q}}\Big) \Big| + \Big| (B_t^{\tilde{r}})^{-1}\Big(\Psi(\widehat{V}_t) - \Psi(\widehat{V}_s)\Big)\lambda_s^{C,\mathbb{Q}} \Big| + \Big| \Big((B_t^{\tilde{r}})^{-1} - (B_s^{\tilde{r}})^{-1}\Big)\Psi(\widehat{V}_s)\lambda_s^{C,\mathbb{Q}} \Big| \Big]$$

$$\leq \; C\left\{\mathbb{E}[(\lambda_t^{C,\mathbb{Q}} - \lambda_s^{C,\mathbb{Q}})^2]^{1/2} + \mathbb{E}[(\widehat{V}_t - \widehat{V}_s)^2]^{1/2} + \mathbb{E}\left[\left(1 - \exp\left(-\int_s^t \tilde{r}_u\, \mathrm{d}u\right)\right)^2\right]^{1/2}\right\},$$

for some constant C, using the boundedness of r, $\lambda^{C,\mathbb{Q}}$, $\lambda^{B,\mathbb{Q}}$, and of $\mathbb{E}[(\Psi(\widehat{V}_t))^2]$. The first and last term are of order $|t - s|^{1/2}$ by the assumptions made, and it remains to estimate the middle term. Recalling that $\widehat{V}_t = \sum_{m=1}^M \widehat{V}_t^m$ with $\widehat{V}_t^m$ the solution of the FBSDE (2.1), (2.6), under the regularity assumptions on the coefficients μ and σ of the forward SDE one gets

$$\left|\varphi(t) - \varphi(s)\right| \leq C\left(|t - s|^{1/2} + \sum_{m=1}^M \mathbb{E}\left[|\widehat{V}_t^m - \widehat{V}_s^m|^2\right]^{1/2}\right)$$

$$\leq C\left(|t - s|^{1/2} + \sum_{m=1}^M \mathbb{E}\left[|g_m(S_T)|^2 |t - s| + \int_s^t |\widehat{Z}_u^m|^2\, \mathrm{d}u\right]^{1/2}\right)$$

$$\leq C\left(|t - s|^{1/2} + |t - s|^{1/2} \sum_{m=1}^M \mathbb{E}\left[|g_m(S_T)|^2 + 1 + \sup_{u\in[s,t]} |S_u|^2\right]^{1/2}\right)$$

$$\leq C|t - s|^{1/2}.$$

To obtain the estimates for the increment of the BSDE solution and of the control in terms of the forward process, in the second and third line, respectively, we can apply Zhang (2004, Lemma 2.4, (2.11)) and Zhang (2017, Theorem 5.2.2(i)) to the equivalent BSDE

(B.2) $\mathrm{d}\widetilde{V}_t^m = \widetilde{Z}_t^m\, \mathrm{d}W_t^{\mathbb{Q}}, \quad \widetilde{V}_T^m = g_m(S_T), \qquad \text{where} \quad \widetilde{V}_t^m = \widehat{V}_t^m B_T^r/B_t^r, \quad \widetilde{Z}_t^m = \widehat{Z}_t^m B_T^r/B_t^r.$

Above and in the following, we do not keep track of constants and C denotes any non-negative constant depending only on T, M, s_0 and the regularity constants of the coefficients.

Then, if we consider the rectangle quadrature rule we get

(B.3) $Q(\Delta t) \leq C\Delta t^{1/2}.$

Therefore, observing that

$$\left(\sum_{n=0}^N |\eta_n|^2\right)^{1/2} \leq \Delta t N^{1/2}$$

and using (B.1) and (B.3), one has

$$\left|\mathbb{E}\left[\int_0^T \Phi(t, \widehat{V}_t)\, \mathrm{d}t\right] - \mathbb{E}\left[\sum_{n=0}^{N-1} \Delta t \Phi(t_n, \widehat{V}_{t_n}^{\xi,\rho})\right]\right|$$

$$\leq C\Delta t^{1/2} + C\Delta t N^{1/2} N^{1/2}\left(\sup_{n=0,\dots,N-1} \mathbb{E}\left[|\widehat{V}_{t_n} - \widehat{V}_{t_n}^{\xi,\rho}|^2\right]\right)^{1/2}$$

$$\leq C\Delta t^{1/2} + TC\left(\Delta t + \sum_{m=1}^M \mathbb{E}\left[|g_m(S_T) - \widehat{V}_T^{m,\xi_m,\rho_m}|^2\right]\right)^{1/2},$$

from which the claim (3.15) follows just taking $\xi = (\xi_1^*, \dots, \xi_M^*)$ and $\rho = (\rho_1^*, \dots, \rho_M^*)$.

Appendix C. Hyperparameters tuning

The aim of this section is to test the response of our algorithm with respect to changes in the hyperparameters. In particular, we will study the behaviour of the solution and of the loss function with respect to the variation of the number of iterations $\mathcal{I}$, the number of hidden layers $\mathcal{L} - 1$ and the value of the learning rate used in the stochastic gradient descent algorithm.

We focus on the cases of the forward contract and the call option for which we can provide exact

solutions as benchmarks. Table 5 shows the impact of increasing the number of iterations on the loss function. We observe a substantial reduction of the loss as the number of iterations increases from 100 to 10000. We point out that the increse in the number of iterations has to be coupled with a suitable schedule of the learning rate. The computational time of course increases, showing that the reduction of the loss comes at a non-negligible cost in terms of computational time. In Figures 9 and 10 we can qualitatively observe that the exposure profile produced by the solver is in good agreement with the analytical one even for $\mathcal{I} = 500$, a relatively low number of iterations. In the numerical tests in Section 4 we used $\mathcal{I} = 4000$.

$\mathcal{I}$	loss	CPU(s)		$\mathcal{I}$	loss	CPU (s)
100	4.6802e+02	103		100	2.8109e+02	144
200	2.1080e+01	135		200	5.7360e+01	157
500	4.3487e+00	163		500	1.2659e+01	241
1000	2.9029e+00	283		1000	1.1770e+01	284
4000	1.6640e-01	503		4000	7.7790e-01	543
10000	3.2899e-02	765		10000	5.6281e-01	798
40000	6.7284e-03	3194		40000	4.9823e-01	3267
60000	5.5881e-03	4874		60000	4.0948e-01	5124

TABLE 5. Variation in the loss function for different values of the number $\mathcal{I}$ of iterations for the forward contract (left) and the call option (right). Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L}-1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, time steps $N = 200$

The next investigations involve the architecture of the network.   We firstly progressively increased the number of hidden layers from 1 up to 8. We performed the test for the call option since in the case of the forward contract a single layer is sufficient to provide a very good fit. The results are reported in Table 6. We experienced a moderate reduction of the loss when increasing the number of hidden layers beyond 2. The loss can be further reduced by coupling the increase of the number of layers with a higher number of iterations (see right-hand table). Secondly, we increased the number of nodes from $d + 20 = 21$ to $d + 200 = 201$, fixing 2 hidden layers. The results reported in Table 7 show again a small reduction of the loss. We tested two different numbers of iterations ($\mathcal{I} = 4000$ and $\mathcal{I} = 40000$, top row) and batch sizes ($B = 64$ and $B = 256$, bottom row), obtaining a further loss reduction from increasing $\mathcal{I}$ and B.

$\mathcal{L} - 1$	loss	CPU (s)		$\mathcal{L} - 1$	loss	CPU (s)
1	2.5790e+00	356		1	2.1964e+00	2053
2	7.7790e-01	543		2	4.9823e-01	3267
4	6.3602e-01	781		4	4.8404e-01	5388
8	6.8549e-01	1486		8	4.2942e-01	10434

TABLE 6. Call option. Variation of the loss function for different number $\mathcal{L} - 1$ of hidden layers. Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, nodes of each internal layer $\nu = d + 20 = 21$, time steps $N = 200$, iterations $\mathcal{I} = 4000$ (left) and $\mathcal{I} = 40000$ (right).
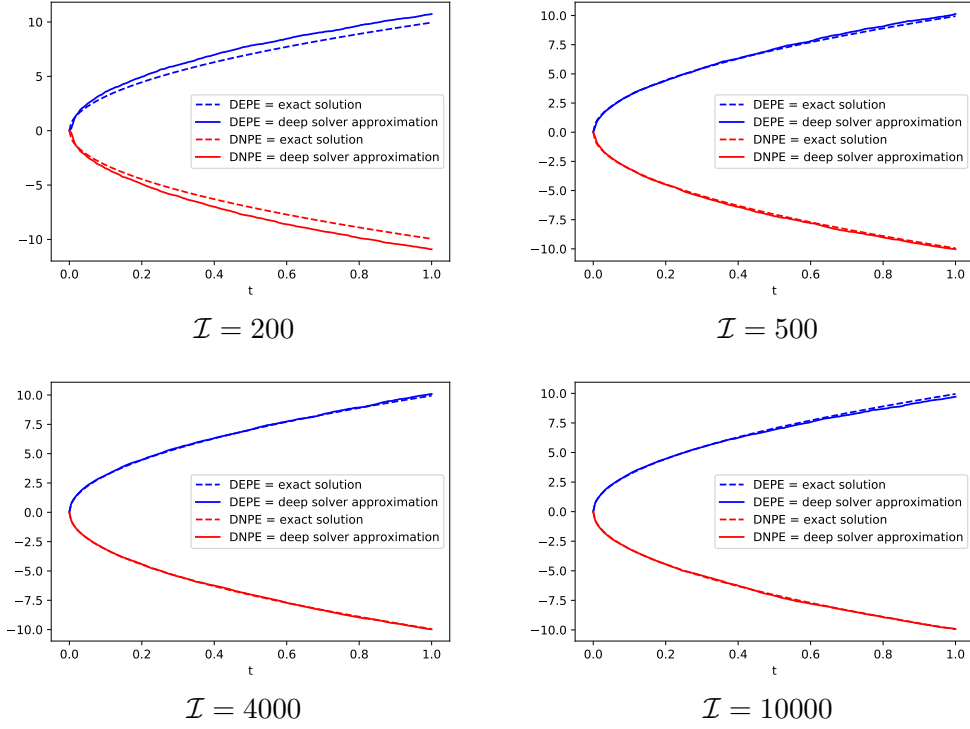
FIGURE 9. Exposure profile for the forward contract for different numbers of iterations. Parameters used: outer MC paths $P = 10000$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, time steps $N = 200$

ν	loss	CPU (s)
21	7.7790e-01	543
201	6.4578e-01	1741

$B = 64, \mathcal{I} = 4000$

ν	loss	CPU (s)
21	4.9823e-01	3267
201	4.2650e-01	10425

$B = 64, \mathcal{I} = 40000$

ν	loss	CPU (s)
21	5.6281e-01	798
201	5.0817e-01	2922

$B = 64, \mathcal{I} = 10000$

ν	loss	CPU (s)
21	4.0739e-01	1481
201	3.5423e-01	9654

$B = 256, \mathcal{I} = 10000$

TABLE 7. Call option. Variation of the loss function for different number of nodes ν. Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, time steps $N = 200$, internal layers $\mathcal{L} - 1 = 2$. Top row: batch size $B = 64$, iterations $\mathcal{I} = 4000$ (left) and $\mathcal{I} = 40000$ (right). Bottom row: batch size $B = 64$ (left) and $B = 256$ (right), iterations $\mathcal{I} = 10000$.

We then study the impact of changing the learning rate. Remember that the learning rate represents the step size of the parameters updates during the training procedure. In our algorithm, we employ a so-called *step decay*, meaning that the learning rate is piecewise constant over the number of iterations. In the next experiments, we use a learning rate schedule of the form $[lr1, lr2]$, with two values $lr1, lr2$, where $lr1$ is used for the first half of the iterations, whereas $lr2$ is used for the remaining iterations. From the tests reported in Table 8 and Table 9, we deduce that the choice of a too high learning rate may negatively affect the results, while for low learning rates a higher number of iterations is required
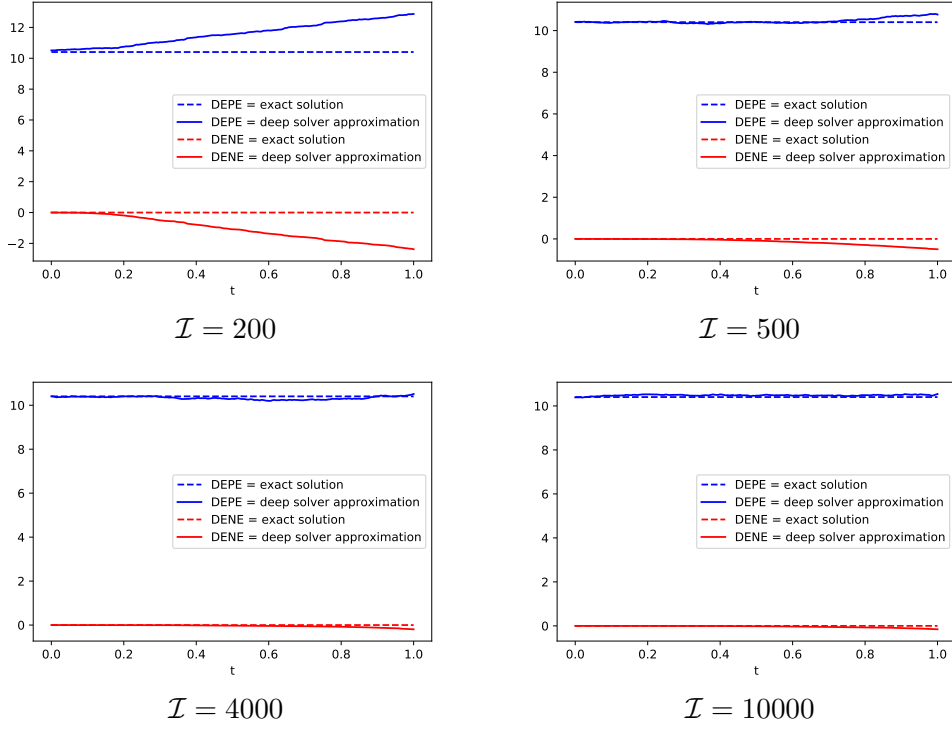
FIGURE 10. Exposure profile for the call option for different numbers of iterations. Parameters used: outer MC paths $P = 10000$, inner MC paths $L = 1024$, batch size $B = 64$, internal layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, time steps $N = 200$

to ensure sufficient training of the network. In the numerical tests in Section 4 we have chosen the schedule $[5e - 2, 5e - 3]$, which was generally observed to provide a good fit both for a high and a low number of iterations.

Finally, Figure 11 shows the convergence in terms of time discretization and batch size for the call option. We observe that increasing the batch size improves the convergence behavior. When the batch size reaches 256, the error levels off so that a further increase does not give a significant improvement, keeping in mind also the associated increase in the execution time of the algorithm. The right panel shows the effect of refining the time discretization. We observe that, as we increase the number of time steps up to 200, the numerical error is reduced, while further increases do not bring further reductions of the loss. This is possibly due to the significant increase in the number of neural network parameters to be estimated (one network per timestep), while $\mathcal{I}$ and L are not modified in these tests.
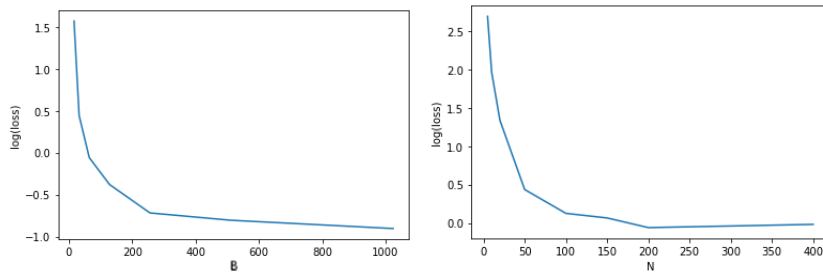

FIGURE 11. Call option. Left: convergence with respect to batch size B, parameters used: iterations $\mathcal{I} = 4000$, time steps $N = 200$. Right: convergence with respect to the number of time steps N, parameters used: iterations $\mathcal{I} = 4000$, batch size $B = 64$. .

l.r.	loss	CPU(s)
[1e-3,1e-4]	4.7649e+01	469
[5e-3,5e-4]	1.5160e+00	445
[1e-2, 1e-3]	5.6159e-01	476
[5e-2,5e-3]	1.6640e-01	503
[1e-1,1e-2]	1.7459e-01	423
[5e-1,5e-2]	1.3840e+01	434

l.r.	loss	CPU (s)
[1e-3,1e-4]	1.1080e+01	392
[5e-3,5e-4]	8.2645e-01	399
[1e-2, 1e-3]	6.1086e-01	462
[5e-2,5e-3]	7.7790e-01	543
[1e-1,1e-2]	1.0811e+00	498
[5e-1,5e-2]	1.0183e+01	420

TABLE 8. Effects of changes of the learning rate for the forward contract (left) and the call option (right). Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, iterations $\mathcal{I} = 4000$, number of hidden layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, time steps $N = 200$. The first value of the learning rate is used for the first 2000 iterations.

l.r.	loss	CPU(s)
[1e-3,1e-4]	1.2445e+00	1523
[5e-3,5e-4]	4.3438e-02	1539
[1e-2, 1e-3]	1.0684e-01	1565
[5e-2,5e-3]	7.8567e-02	1516
[1e-1,1e-2]	1.6278e-01	1666
[5e-1,5e-2]	1.4697e+01	1691

l.r.	loss	CPU (s)
[1e-3,1e-4]	6.4950e-01	1769
[5e-3,5e-4]	5.6589e-01	1815
[1e-2, 1e-3]	6.0227e-01	1903
[5e-2,5e-3]	6.6738e-01	1767
[1e-1,1e-2]	8.9118e-01	1732
[5e-1,5e-2]	5.1818e+01	1824

TABLE 9. Effects of changes of the learning rate for the forward contract (left) and the call option (right). Parameters used: outer MC paths $P = 2048$, inner MC paths $L = 1024$, batch size $B = 64$, iterations $\mathcal{I} = 20000$, number of hidden layers $\mathcal{L} - 1 = 2$, nodes of each internal layer $\nu = d + 20 = 21$, time steps $N = 200$. The first value of the learning rate is used for the first 10000 iterations.

REFERENCES

Abbas-Turki, L. A., Crépey, S., and Diallo, B. (2018). XVA principles, nested Monte Carlo strategies, and GPU optimizations. International Journal of Theoretical and Applied Finance, 21(6):1850030.

Albanese, C., Caenazzo, S., and Crépey, S. (2017). Credit, funding, margin, and capital valuation adjustments for bilateral portfolios. Probability, Uncertainty and Quantitative Risk, 2(1).

Albanese, C., Crépey, S.and Hoskinson, R., and Saadeddine, B. (2021). XVA analysis from the balance sheet. Quantitative Finance, 21(1):99–123.

Andersen, L., Duffie, D., and Song, Y. (2019). Funding value adjustments. The Journal of Finance, 74(1):145–192.

Bender, C. and Steiner, J. (2013). A posteriori estimates for backward SDEs. SIAM/ASA Journal on Uncertainty Quantification, 1(1):139–163.

Biagini, F., Gnoatto, A., and Oliva, I. (2021). A unified approach to xVA with CSA discounting and initial margin. Siam Journal on Financial Mathematics - Accepted.

Bichuch, M., Capponi, A., and Sturm, S. (2018a). Arbitrage-free XVA. Mathematical Finance, 28(2):582–620.

Bichuch, M., Capponi, A., and Sturm, S. (2018b). Robust XVA. arXiv preprint arXiv:1808.04908.

Bielecki, T. and Rutkowski, M. (2004). Credit Risk: Modeling, Valuation and Hedging. Springer, Berlin, New York.

Bielecki, T. and Rutkowski, M. (2015). Valuation and hedging of contracts with funding costs and collateralization. SIAM Journal of Financial Mathematics, 6(1):594–655.

Borovykh, A., Pascucci, A., and Oosterlee, C. W. (2018). Efficient computation of various valuation adjustments under local Lévy models. SIAM Journal on Financial Mathematics, 9(1):251–273.

Brigo, D., Buescu, C., Francischello, M., Pallavicini, A., and Rutkowski, M. (2018). Risk-neutral valuation under differential funding costs, defaults and collateralization. arXiv preprint arXiv:1802.10228.

Brigo, D., Capponi, A., and Pallavicini, A. (2014). Arbitrage-free bilateral counterparty risk valuation under collateralization and application to credit default swaps. Mathematical Finance, 24(1):125–146.

Brigo, D., Francischello, M., and Pallavicini, A. (2019). Nonlinear valuation under credit, funding and margins: existence uniqueness, invariance and disentanglement. European Journal of Operational Research, 274(2):788–805.

Brigo, D. and Masetti, M. (2005). Risk neutral pricing of counterparty risk. In Pykhtin, M., editor, Counterparty Credit Risk Modeling: Risk Management, Pricing and Regulation. Risk Books, London.

Brigo, D. and Pallavicini, A. (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, 01(01):1450001.

Brigo, D., Pallavicini, A., and Papatheodorou, V. (2011). Arbitrage-free valuation of bilateral counterparty risk for interest-rate products: Impact of volatilities and correlations. International Journal of Theoretical and Applied Finance, 14(06):773–802.

Broadie, M., Du, Y., and Moallemi, C. (2011). Efficient risk estimation via nested sequential simulation. Management Science.

Broadie, M., Du, Y., and Moallemi, C. (2015). Risk estimation via regression. Operations Research.

Burgard, C. and Kjaer, M. (2011a). In the balance. Risk, November:72–75.

Burgard, C. and Kjaer, M. (2011b). Partial differential equation representations of derivatives with bilateral counterparty risk and funding costs. The Journal of Credit Risk, 7(3):1–19.

Capriotti, L., Jiang, Y., and Macrina, A. (2017). AAD and least-square Monte Carlo: Fast Bermudan-style options and XVA Greeks. Algorithmic Finance, 6(1-2):35–49.

Cherubini, U. (2005). Counterparty risk in derivatives and collateral policies: The replicating portfolio approach. In Tilman, L., editor, ALM of Financial Institutions. Institutional Investor Books.

Crépey, S. (2015a). Bilateral counterparty risk under funding constraints – Part I: Pricing. Mathematical Finance, 25(1):1–22.

Crépey, S. (2015b). Bilateral counterparty risk under funding constraints – Part II: CVA. Mathematical Finance, 25(1):23–50.

Crépey, S. and Dixon, M. (2020). Gaussian process regression for derivative portfolio modeling and application to CVA computations. Journal of Computational Finance, 24(1):47–81.

Cuchiero, C., Fontana, C., and Gnoatto, A. (2019). Affine multiple yield curve models. Mathematical Finance, 29(2):568–611.

Cybenko, G. (1989). Approximations by superpositions of sigmoidal functions. Mathemtics of Control, Signals, and Systems, 2(4):303–314.

de Graaf, C., Feng, Q., Kandhai, B. D., and Oosterlee, C. (2014). Efficient computation of exposure profiles for counterparty credit risk. International Journal of Theoretical and Applied Finance,

17(4).

de Graaf, C., Kandhai, B. D., and Reisinger, C. (2018). Efficient exposure computation by risk factor decomposition. Quantitative Finance, 18(10):1657–1678.

Delong, L. (2017). Backward Stochastic Differential Equations with Jumps and Their Actuarial and Financial Applications. Springer, Berlin, New York.

Duffie, D. and Huang, M. (1996). Swap rates and credit quality. The Journal of Finance, 51(3):921–949.

El Karoui, N., Peng, S., and Quenez, M. C. (1997). Backward stochastic differential equations in finance. Mathematical Finance, 7(1):1–71.

Fang, F. and Oosterlee, C. (2009). A novel pricing method for European options based on Fourier-cosine series expansions. SIAM Journal on Scientific Computing, 31(2):826–848.

Ferguson, R. and Green, A. (2018). Deeply learning derivatives. arXiv preprint arXiv:1809.02233.

Fries, C. P. (2019). Stochastic automatic differentiation: Automatic differentiation for Monte-Carlo simulations. Quantitative Finance, 19(6):1043–1059.

Fujii, M., Shimada, A., and Takahashi, A. (2010). Note on construction of multiple swap curves with and without collateral. Available at SSRN:http://ssrn.com/abstract=1440633.

Fujii, M., Shimada, A., and Takahashi, A. (2011). A market model of interest rates with dynamic basis spreads in the presence of collateral and multiple currencies. Wilmott, 54:61–73.

Fujii, M., Takahashi, A., and Takahashi, M. (2019). Asymptotic expansion as prior knowledge in deep learning method for high dimensional BSDEs. Asia-Pacific Financial Markets, 26(3):391–408.

Gnoatto, A. and Seiffert, N. (2021). Cross currency valuation and hedging in the multiple curve framework. Siam Journal on Financial Mathematics - Accepted.

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

Han, J., Jentzen, A., and E, W. (2018). Solving high-dimensional partial differential equations using deep learning. Proceedings of the National Academy of Sciences, 115(34):8505–8510.

Han, J. and Long, J. (2020). Convergence of the deep BSDE method for coupled FBSDEs. Probability, Uncertainty and Quantitative Risk, 5(1):1–33.

Henry-Labordere, P. (2017). Deep primal-dual algorithm for BSDEs: applications of machine learning to CVA and IM. Available at SSRN:https://ssrn.com/abstract=3071506.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. Neural Networks, 4(2):251–257.

Huré, C., Pham, H., and Warin, X. (2020). Some machine learning schemes for high-dimensional nonlinear PDEs. Mathematics of Computations, 89:1547–1579.

Hutzenthaler, M., Jentzen, A., Kruse, T., Nguyen, T. A., and von Wurstemberger, P. (2018). Overcoming the curse of dimensionality in the numerical approximation of semilinear parabolic partial differential equations. arXiv preprint arXiv:1807.01212.

Hutzenthaler, M., Jentzen, A., and Noll, M. (2014). Strong convergence rates and temporal regularity for Cox-Ingersoll-Ross processes and Bessel processes with accessible boundaries. arXiv preprint arXiv:1403.6385.

Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proceeding of the 32nd International Conference on Machine Learning (CML).

Jentzen, A., Salimova, D., and Welti, T. (2018). A proof that deep artificial neural networks overcome the curse of dimensionality in the numerical approximation of Kolmogorov partial differential equations with constant diffusion and nonlinear drift coefficients. arXiv preprint arXiv:1809.07321.

Joshi, M. and Kwon, O. (2016). Least squares Monte Carlo credit value adjustment with small and unidirectional bias. International Journal of Theoretical and Applied Finance, 19(8).

Karlsson, P., Jain, S., and Oosterlee, C. (2016). Counterparty credit exposures for interest rate derivatives using the stochastic grid bundling method. Applied Mathematical Finance, 23(3):175–196.

Kolmogorov, A. N. (1956). On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. Doklady Akademii Nauk SSSR, 108(2):679–681.

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

Ninomiya, S. and Shinozaki, Y. (2019). Higher-order discretization methods of forward-backward SDEs using KLNV-scheme and their applications to XVA pricing. Applied Mathematical Finance, 26(3):257–292.

Pallavicini, A., Perini, D., and Brigo, D. (2011). Funding Valuation Adjustment: a consistent framework including CVA, DVA, collateral, netting rules and re-hypothecation. arXiv preprint arXiv:1112.1521.

Piterbarg, V. (2010). Funding beyond discounting: collateral agreements and derivatives pricing. Risk Magazine, 2:97–102.

Piterbarg, V. (2012). Cooking with collateral. Risk Magazine, 2:58–63.

Reisinger, C., Stockinger, W., and Zhang, Y. (2020). A posteriori error estimates for fully coupled McKean-Vlasov forward-backward SDEs. arXiv preprint arXiv:2007.07731.

Reisinger, C. and Zhang, Y. (2020). Rectified deep neural networks overcome the curse of dimensionality for nonsmooth value functions in zero-sum games of nonlinear stiff systems. Analysis and Applications. https://doi.org/10.1142/S0219530520500116.

Ruf, J. and Wang, W. (2019). Neural networks for option pricing and hedging: a literature review. Available at SSRN:3486363.

She, J.-H. and Grecu, D. (2017). Neural network for CVA: Learning future values. arXiv preprint arXiv:1811.08726.

Shöftner, R. (2008). On the estimation of credit exposures using regression-based Monte Carlo simulation. The Journal of Credit Risk, 4(4):37–62.

Zhang, J. (2004). A Numerical Scheme for BSDEs. The Annals of Applied Probability, 14(1):459–488.

Zhang, J. (2017). Backward Stochastic Differential Equations. Springer, New York.

(Alessandro Gnoatto) University of Verona, Department of Economics,
via Cantarane 24, 37129 Verona, Italy
Email address, Alessandro Gnoatto: `alessandro.gnoatto@univr.it`

(Athena Picarelli) University of Verona, Department of Economics,
via Cantarane 24, 37129 Verona, Italy
Email address, Athena Picarelli: `athena.picarelli@univr.it`

(Christoph Reisinger) Oxford University, Mathematical Institute
ROQ, Woodstock Rd, Oxford, OX2 6GG, UK
Email address, Christoph Reisinger: `christoph.reisinger@maths.ox.ac.uk`