Learning From Simulated Data in Finance : XVAs, Risk Measures and Calibration

Bouazza Saadeddine

université PARIS-SACLAY

Learning From Simulated Data in Finace : XVAs, Risk Measures and Calibration

Apprentissage sur données simulées en finance : XVAs, mesures de risque et calibration

Thèse de doctorat de l'université Paris-Saclay

École doctorale n°574, École Doctorale de Mathématiques Hadamard (EDMH)
Spécialité de doctorat : Mathématiques appliquées
Graduate School : Mathématiques
Référent : Université d'Evry Val d'Essonne

Thèse préparée dans l'unité de recherche **Université Paris-Saclay, Univ Evry, Laboratoire de Mathématiques et Modélisation d'Evry, CNRS UMR 8001, Evry, France**,
sous la direction de **Stéphane CRÉPEY**, Professeur des Universités,
le co-encadrement de **Lokman ABBAS-TURKI**, Maître de Conférences,
et la co-supervision de **Christophe MICHEL**, Responsable Rech. Quantitative à CACIB

Thèse soutenue à Paris, le 21 octobre 2022, par

Bouazza SAADEDDINE

Composition du jury

Huyên PHAM Professeur des universités, Université Paris-Cité	Président
Pierre HENRY-LABORDÈRE Global head of quantitative research, Natixis	Rapporteur & Examinateur
Josef TEICHMANN Professor, ETH Zürich	Rapporteur & Examinateur
Ludovic GOUDENÈGE Chargé de recherche CNRS, CentraleSupélec	Examinateur
Blanka HORVATH Professor, University of Oxford	Examinatrice
Olivier PIRONNEAU Professeur émérite, Sorbonne Université	Examinateur
Stéphane CRÉPEY Professeur des universités, Université Paris-Cité	Directeur de thèse

THESE DE DOCTORAT
NNT : 2022UPASM024

Acknowledgements

I would firstmost like to acknowledge and give my sincerest thanks to my PhD supervisors Stéphane Crépey and Lokmane Abbas-Turki for their trust, guidance and constant support during my thesis which made this work possible. I am also grateful to Stéphane for having presented me the opportunity of this CIFRE thesis with Crédit Agricole CIB (CACIB) when I was still hesitating on my career path towards the end of my Masters and for having always supported me. Chapter 5, in particular, would probably not have seen the light of the day without his encouragement and support. I also thank Lokmane for sharing his passion and expertise in numerical finance with CUDA with me.

I am also thankful to CACIB and ANRT for financing this thesis. I am also grateful to Christophe Michel and Vincent Porte for having made this CIFRE thesis possible and for welcoming me into CACIB where they were always available for technical and financial discussions around my works, of which Chapter 5 is a direct result for instance. I am also grateful to Moez Mrad for insightful discussions around XVA calculations and hedging. My thanks extend to my quant and IT colleagues who made my three years at CACIB a real joy.

I sincerely thank Pierre Henry-Labordère and Joseph Teichmann for honouring me by accepting to review my manuscript. I am equally honoured by having Ludovic Goudnège, Blanka Horvath, Huyên Pham and Olivier Pironneau in my jury. I am also grateful to Joseph Teichmann and Blanka Horvath for having traveled from Switzerland and the United Kingdom to attend my thesis defense in-person.

My thesis also would not have been successful without the help and availability of the LaMME and LPSM staff. In this regard, I would like to thank Patricia Authier and Maurice Baudry for allowing me access to the laboratories' GPU servers, which were vital for my computational results, and for always being there to help me with server-related issues. I would also like to thank Nathalie Bergame, Valérie Gontier-Picot and Elise Maspimby for welcoming me into the laboratories and helping me with all the administrative tasks.

I also wish to extend my gratitude to the Graduate School of Mathematics Hadamard and especially Ana-Maria Castravet, Pierre Gilles Lemarié-Rieusset, Stéphane Menozzi, Stéphane Nonnenmacher, Vincent Sécherre for their crucial follow-ups during my thesis and for having made the organization of the defense a success.

I also thank Michael Allouche, Emmanuel Gobet, Zoltan Szabo and Elodie Vernet for organizing the Machine Learning Journal Club at CMAP and for inviting me to speak there at multiple occasions.

I would also like to thank everyone at CMAP, LaMME and LPSM with whom I had the chance to work or have fruitful discussions, in particular David Barrera, Cyril Benezet, Mohamed-Raed Blel, Marc Chataignier, Babacar Diallo and Hoang Dung Nguyen.

This manuscript was written entirely in TeXmacs (texmacs.org) and I am grateful for the help provided by the community at forum.texmacs.cn which made writing this manuscript a breeze.

Finally, none of my accomplishments would have come forward without the constant love and support of my family. I am hereby forever grateful to my parents and to my brother AZZAD.

Table of contents

Introduction

1 Context

The mathematical notations in this section are local to this section only.

1.1 The Era of Big Data and Machine Learning

The 21st century is marked by an abundance of data [Sirko et al., 2021; Sidorov et al., 2020; Kuznetsova et al., 2020; Abu-El-Haija et al., 2016; Deng et al., 2009], the emergence of new technologies to structure and manage it [Zaharia et al., 2016; Lakshman and Malik, 2010; Borthakur, 2007], and continuous hardware innovations to process it in reasonable time [Nickolls and Dally, 2010]. This defined the so-called era of *"Big Data"* [Magoulas and Lorica, 2009] marked by the ubiquity of parallel programming techniques and architectures, the widespread use of massively parallel compute architectures such as Graphics Processing Units (GPUs) via programming APIs such as CUDA [Luebke, 2008] and OpenCL [Stone et al., 2010] due to the reliance of most state-of-the-art algorithms on linear algebra routines that benefit greatly from GPU acceleration [Shi et al., 2016], and the prevalence of stochastic programming algorithms, from the original version of stochastic gradient descent (SGD) studied in the seminal work of Robbins and Monro, 1951 to the most recent and immensely successful variants such as the ADAM algorithm by Kingma and Ba, 2014, due to the attractive scalability properties of these algorithms [Bottou, 2010].

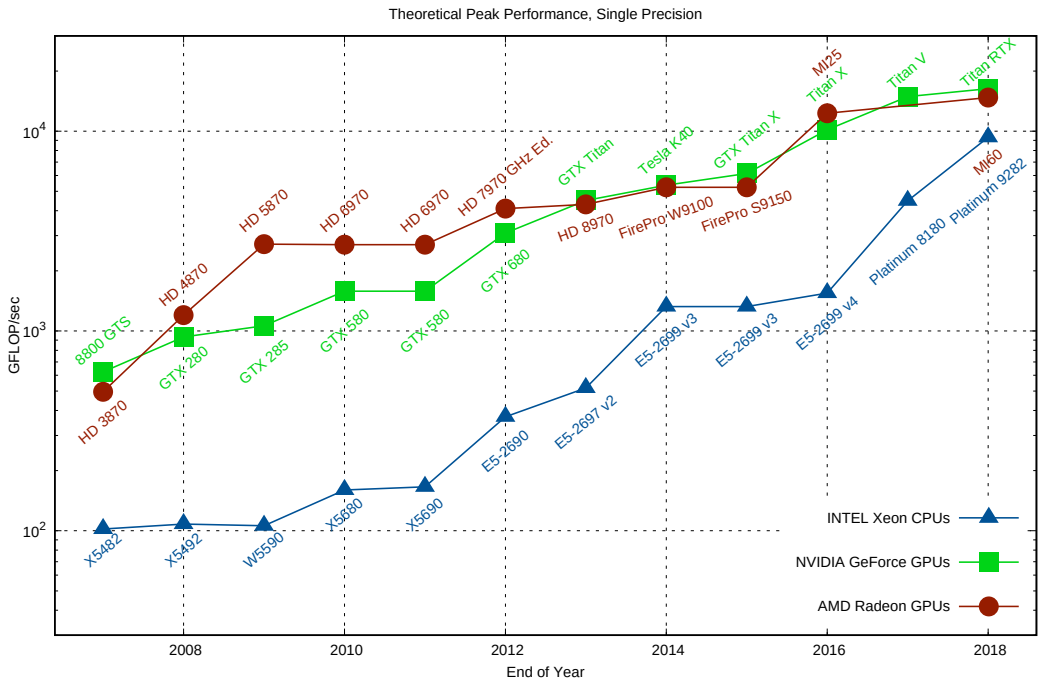

Figure 1. Theoretical peak floating-point operations per second (FLOP/s, 1 GFLOP/s=10^9 FLOP/s) for Intel CPUs vs AMD and NVidia GPUs in simple precision. Produced using code and data by Karl Rupp available at `https://github.com/karlrupp/cpu-gpu-mic-comparison` under a CC BY 4.0 license.

With this revolution in compute architectures, parallel programming, stochastic optimization and other scalable numerical methods, another paradigm to designing computer programs gained in prominence, where programmers do not have to explicitly program the solution to a problem they are trying to solve and where they can instead implement an algorithm which *learns* the solution, or at least how to get close to one, given sufficient amounts of data by using statistical learning techniques. This new paradigm is more relevant now than ever in this era of *Big Data*. The essence of this paradigm can perhaps be best summarized by the following quotes from Samuel, 1959, who coined the term *Machine Learning*, in the context of the game of checkers:

> [...] *a computer can be programmed so that it will* ***learn*** *to play a better game of checkers than can be played by the person who wrote the program. Furthermore, it can learn to do this in a remarkably short period of time.* [...] *Programming computers to* ***learn from experience*** *should eventually eliminate the need for much of this detailed programming effort.*

Machine Learning approaches, and more prevalently Neural Network based ones, knew tremendous success in disciplines such as computer vision [Janai et al., 2020; Voulodimos et al., 2018], natural language processing [Wolf et al., 2019; Young et al., 2018], recommender systems [Zhang et al., 2019; Adomavicius and Tuzhilin, 2005] or anomaly detection [Chalapathy and Chawla, 2019; Zenati et al., 2018; Omar et al., 2013].

Although there was a certain adoption by financial institutions, mostly in consumer banking with applications such as credit scoring [Lessmann et al., 2015] or fraud detection [Chan et al., 1999], its use remained limited in quantitative finance especially on the sell-side. This is, in our view, mostly due to a common misconception that Machine Learning would apply only on problems with real-world applications and empirical data, as opposed to mathematical models and risk-neutral valuation frameworks in pricing and hedging problems. Applications emerged nevertheless especially in *fast pricing* applications [Horvath et al., 2021; De Spiegeleer et al., 2018] where the goal is to construct fast price approximations that could replace slow pricing routines, *e.g.* based on Monte Carlo simulations. Goudenege et al., 2020 propose to combine a one-step tree method [Ekvall, 1996] with Gaussian process regression [Rasmussen and Williams, 2006] in order to approximate the continuation value at exercise dates when valuing a Bermudan option via backward dynamic programming. These applications are undoubtedly essential as they are important building blocks on which transactions, regulatory and risk calculations, model calibration and electronic trading, among others, are highly dependent both functionally and performance-wise, *i.e.* fast pricing libraries necessarily translate into faster risk calculations. In these settings, Machine Learning algorithms are employed in explicit function fitting setups.

New uses of Machine Learning specific to mathematical finance models began to emerge thanks to seminal works such as [Huré et al., 2020; Raissi, 2018; Han et al., 2018]. By casting *backward stochastic differential equations* (BSDEs) [Pardoux and Peng, 1990] as stochastic control problems in discrete time, the authors manage to solve BSDEs or the associated *partial differential equations* (PDEs) using a space of neural networks with a suitable architecture as their search space in the minimization problem and paths simulated via Monte Carlo as their *training data*. The applicability of BSDEs in finance was first highlighted in [El Karoui et al., 1997] and a more up-to-date treatment is given in [Crépey, 2013]. Buehler et al., 2019 use a formally similar approach to directly tackle the problem of hedging a financial derivative product in the presence of market imperfections like

transaction costs, slippage and market impact, which are usually neglected in common risk-neutral valuation frameworks. Becker et al., 2019 also propose a novel technique for solving optimal stopping problems by representing the policy using neural networks and show applications in pricing Bermudan and callable derivatives. This marked the beginning of a trend where Machine Learning in finance goes beyond function fitting and exploited the unique mathematical setting of the models used by the banks.

This thesis also continues along the same line of thought in devising Machine Learning based schemes, or more precisely neural networks based ones, to address the problems of computing *X-Valuation Adjustments* (XVAs) and conditional risk measures and accelerating the calibration of pricing models.

1.2 A Brief XVA Intermezzo

The 2008–2009 financial crisis saw numerous banking reforms aimed at increasing the robustness of the financial system. A consequence of this was an increase in the use of XVA metrics during the pricing of derivative products. These metrics were intended to help quantify and price market incompleteness by banks and account for counterparty risk and its capital and funding implications. The letter X is a catch-all letter to be replaced by:

- C for *credit*, *i.e.* Credit Valuation Adjustment (CVA);
- D for *debt*, *i.e.* Debt Valuation Adjustment (DVA);
- F for *funding*, *i.e.* Funding Valuation Adjustment (FVA);
- M for *margin*, *i.e.* Margin Valuation Adjustment (MVA);
- K for *capital*[1], *i.e.* Capital Valuation Adjustment (KVA).

XVA applications form a sizable part of this thesis due to their importance in regulatory calculations. This is done in the framework of [Albanese et al., 2021] (work which is also presented in Chapter 1), addressing refined features such as the simulation of defaults (see Chapter 2) or the fungibility of the reserve capital and capital at risk with variation margin (see Chapter 4).

For the purpose of this introduction, we will briefly give an example using the CVA and the FVA to highlight the computational complexity involved in the calculation of these metrics. In particular, we address the problem of computing these metrics at future time-steps under a risk-neutral model of market and credit risk-factors.

Indeed, successfully modelling the future evolution of the CVA, and in particular modelling it as a stochastic process, is essential as losses can materialize merely because of the fluctuation of this metric. For instance, the Basel Committee said the following in [The Bank for International Settlements, 2011]:

> *During the financial crisis, roughly two-thirds of losses attributed to counter-*

1. as C was already taken for the CVA.

party credit risk were due to CVA losses and only about one-third were due to actual defaults.

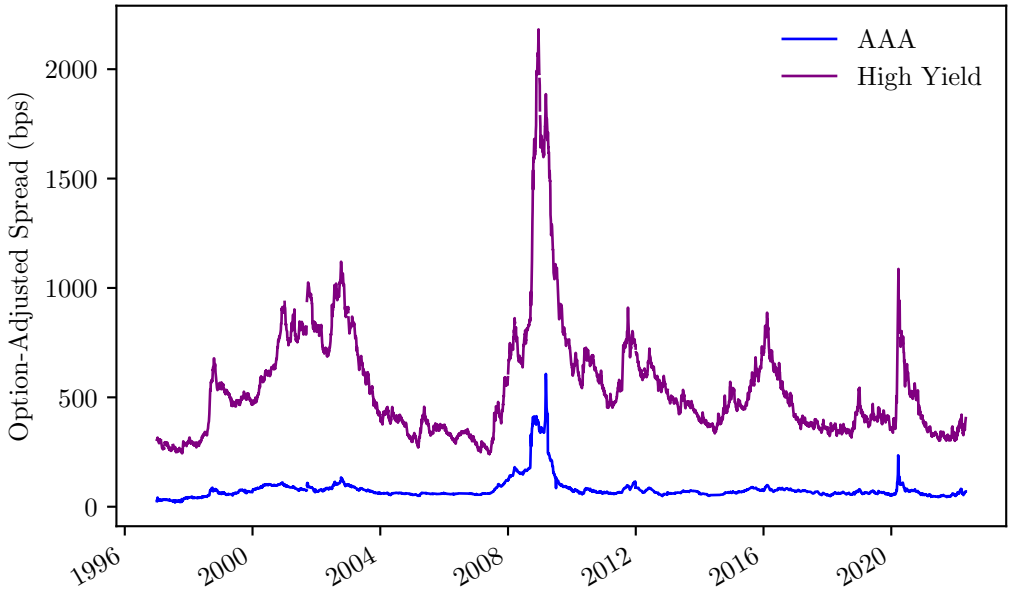

Figure 2. CVA losses can be caused simply by fluctuations in its value caused by changes in credit spreads. This plot represents the ICE BofA US High Yield Index **(purple)** and the ICE BofA AAA US Corporate Index **(blue)** Option-Adjusted Spreads. Data retrieved from FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/BAMLH0A0HYM2 and https://fred.stlouisfed.org/series/BAMLC0A1CAAA.

Assuming for simplicity one single counterparty and one single uncollateralized derivative transaction that the bank and the client entered into, zero recovery in the case of a default, neglecting any discounting, and assuming a time discretization $0 = t_0 < t_1 < \cdots < t_i < \cdots < t_n = T$ where T is the horizon of the product, one can define the CVA process in discrete time as follows:

$$\mathrm{CVA}_i := \mathbb{E}\left[\sum_{j=i}^{n-1} \mathrm{MtM}_{j+1}^{+} \mathbb{1}_{\{t_j < \tau \leq t_{j+1}\}} \middle| X_i, \mathbb{1}_{\{\tau > t_i\}} \right] \tag{1}$$

for each $i \in \{0, \ldots, n-1\}$, where MtM_j is the value, or *mark-to-market*, of the product from the point of view of the bank (*i.e.* positive when it is asset to the bank and negative when it is a liability) at time t_j, τ is the default time of the counterparty, X_i is a vector of market risk factors prevailing at time t_i and we assumed a certain multi-factor pricing model under a stochastic pricing basis. Defined this way, the CVA is an expectation of future losses due to defaults conditional on the prevailing market (through X) and credit (through $\mathbb{1}_{\{\tau > .\}}$)

states. A brute-force method for simulating realizations of the conditional expectation in (1), for example in order to inject them into other nonlinearities or higher-order XVAs like the FVA or simply to compute a Value-at-Risk on it, consists in using an estimator such as:

$$\widehat{\mathrm{CVA}}_i^{(k)} := \frac{1}{M}\sum_{l=1}^{M}\sum_{j=i}^{n-1}\left(\mathrm{MtM}_{j+1}^{(k,l)}\right)^+ \mathbb{1}_{\{t_j<\tau^{(k,l)}\le t_{j+1}\}}$$

for $k\in\{1,\dots,N\}$ where we assume we have access to an i.i.d sample $\left\{\left(X_i^{(k)},\tau^{(k)}\right)\right\}_{1\le k\le N}$ of (X_i,τ) and for each $k\in\{1,\dots,N\}$ we have a conditionally independent sample $\left\{\left(\mathrm{MtM}_{i+1}^{(k,l)},\dots,\mathrm{MtM}_n^{(k,l)},\tau^{(k,l)}\right)\right\}_{1\le l\le M}$ of $(\mathrm{MtM}_{i+1},\dots,\mathrm{MtM}_n,\tau)$ conditional on $(X_i,\mathbb{1}_{\{\tau>t_i\}})=\left(X_i^{(k)},\mathbb{1}_{\{\tau^{(k)}>t_i\}}\right)$. This defines a *Nested Monte Carlo* (NMC) procedure for simulating the conditional expectation in (1). Notice that we would need an additional layer of nested simulations if the MtM is not analytic. Although this brute force scheme can be implemented for the CVA, it becomes out-of-scope for the FVA.

Using the previous notation, and, only for the sake of simplicity of this introduction, neglecting most feedback terms and the fungibility of capital at risk with variation margin [Crépey et al., 2020], we can write for the FVA the following simplified definition:

$$\mathrm{FVA}_i := \mathbb{E}\left[\sum_{j=i}^{n-1}\bar{\gamma}_{j+1}\left(\mathrm{MtM}_{j+1}\mathbb{1}_{\{\tau>t_{j+1}\}}-\mathrm{CVA}_{j+1}-\mathrm{FVA}_{j+1}\right)^+(t_{j+1}-t_j)\middle| X_i,\mathbb{1}_{\{\tau>t_i\}}\right] \quad (2)$$

where $\bar{\gamma}$ is a stochastic process representing the bank's funding spread and is a component in the vector X. Here, the FVA represents the cost of funding our uncollateralized trade and our definition takes into account the fact that the CVA and FVA themselves both help reduce the need for funding. Modelling the FVA as we did also allows one to account for its future fluctuations. Very recently, three US banks suffered a combined $2 billion FVA loss [Becker, 2020] because of how the COVID-19 pandemic, and central banks' responses to it, impacted interest rates and funding spreads (see Table 1).

Tenor	**USD**	**EUR**
2y	226%	398%
5y	119%	170%
10y	83%	69%

Table 1. Jumps in funding spreads from 2020-02-21 to 2020-03-24. Source: Takei, 2020 (IHS Markit).

As one can see from the definition in (2), implementing a purely Nested Monte Carlo procedure for the FVA as we presented for the CVA would suppose to implement nested simulations with as many layers of sub-simulations as there are time steps until maturity because of the dependence on future FVA values in the integrand of the expectation. This of course is prohibitive in terms of computation time and will lead to algorithms that have exponential complexity in the number of time steps. This complexity is further exacerbated when one takes into account other XVAs or risk measures such as the KVA or the economic capital given the coupling they create with the FVA in particular (see Chapter 1).

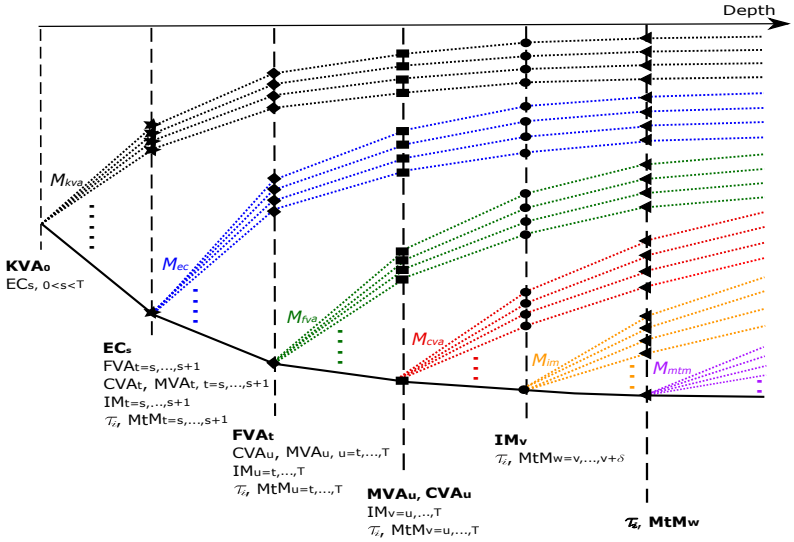

Figure 3. Interdependence between the different XVAs and risk measures in a Nested Monte Carlo context. In such a context, in order to simulate one realization of the economic capital for example at a given Monte Carlo node, one needs inner-simulations of the FVA, CVA and MVA conditional on that node. Each of those *inner FVAs* in turn needs another layer of inner-simulations and so forth. Source: Abbas-Turki et al., 2018.

In [Abbas-Turki et al., 2018], a benchmark approach involving multiple layers of Nested Monte Carlo and linear regressions has been developed along with GPU optimization strategies. That benchmark however has an exponential complexity in the number of XVA layers. In chapters 1, 2 and 4, we develop an approach using Machine Learning, and more precisely regressions based on Neural Networks, in order to make the complexity linear in the number of XVA layers.

1.3 Projections, Not Function Fitting

What the CVA and the FVA, as introduced in (1) and (2) have in common is that each of them is a conditional expectation of a certain integrand. The basic principle behind our proposed approach is to interpret those conditional expectations as orthogonal projections of their respective integrands. More generally, assume we are given two random variables X and Y such that X is supported on a space $\mathcal{X}$ and Y is square integrable and supported on $\mathbb{R}$ and suppose that we are interested in $\mathbb{E}[Y|X]$. Since $\mathbb{E}[Y|X]$ can be seen as an orthogonal projection of Y onto the vector subspace consisting of[2] square integrable $\sigma(X)$-measurable random variables, and hence a minimizer of the associated projection error, one can write:

$$\mathbb{E}[Y|X] = h^\star(X)$$

where $h^\star$ is such that

$$h^\star \in \underset{h\in\mathcal{B}(\mathcal{X},\mathbb{R})}{\operatorname{argmin}}\ \mathbb{E}[(h(X)-Y)^2] \tag{3}$$

where $\mathcal{B}(\mathcal{X},\mathbb{R})$ is the space of Borel functions $h:\mathcal{X}\to\mathbb{R}$ such that $h(X)$ is square integrable. Another way to view this is from a purely probabilistic point of view as follows, where h

2. more precisely, of the equivalence classes with respect to the almost sure equality relation.

is any function in $\mathcal{B}(\mathcal{X}, \mathbb{R})$:

$$\mathbb{E}[(h(X)-Y)^2] = \mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2] + \underbrace{\mathbb{E}[\mathrm{var}(Y|X)]}_{\text{independent of } h}$$

The common strategy then in this thesis is to solve the minimization problem in (3) by introducing two layers of approximation:

- **Approximating** $\mathrm{argmin}_{h\in\mathcal{B}(\mathcal{X},\mathbb{R})}$: we first propose to perform the minimization in a *sufficiently rich* space of parametrized functions so that the minimization becomes finite dimensional and can thus be done with respect to the parameters using classical numerical optimization techniques. Our choice for the entire thesis has been neural networks (we refer for example to chapters 2, 4, 3 or 5 for precise definitions of neural networks as adopted in this thesis) and is motivated by the following considerations:

  - **Universal Approximation property:** If α is a non-affine continuously differentiable activation function, $q \in \mathbb{N}^\star$ and $K \subset \mathbb{R}^n$ a compact. Then for any $\varepsilon > 0$ and $f \in \mathcal{C}(K, \mathbb{R}^q)$, there exists a neural network $u\colon \mathbb{R}^n \to \mathbb{R}^q$ with p hidden layers, $q+n+2$ neurons per hidden layer and α as its activation function such that $\sup_{x\in K} \|u(x) - f(x)\| < \varepsilon$. This result [Kidger and Lyons, 2020] is a *deep* (*i.e.* fixed width) version of the usual fixed depth Universal Approximation theorem for neural networks [Hornik, 1991; Cybenko, 1989]. Hence, one can be optimistic about being able to get close to $h^\star$ by choosing a sufficiently large neural network either in width or in depth, with recent results such as [Cohen et al., 2016; Eldan and Shamir, 2016] more in favor of deep networks;

  - **Ability to automatically *learn* a linear regression basis:** Assuming no nonlinearity at the output of the feed-forward neural network shown in Figure 4, the last hidden layer, when seen as a function Φ_θ of the input x parametrized by the collection θ of all hidden weights, can be interpreted as a parametrized feature map. For fixed θ, denoting by w the weights of the output layer and assuming that the output is scalar, then training a neural network by optimizing only with respect to w amounts to performing a simple linear regression using $\Phi_\theta(x)$ as a feature transform. Thus, training with respect to all weights (*i.e.* θ and w jointly) amounts to learning both the feature map and the associated weights in the output layer. Hence, given enough data, a neural network can automatically learn a linear regression basis and thus avoid the manual selection of regression bases that is usually done by domain experts and which cannot be done for example in the case of XVAs where the dependence of the XVAs on the many risk factors is highly non-trivial. This joins the idea of Samuel, 1959 above that a program can be made to learn how to solve a task, in this case the selection of a regression

basis, eventually better than its programmer;

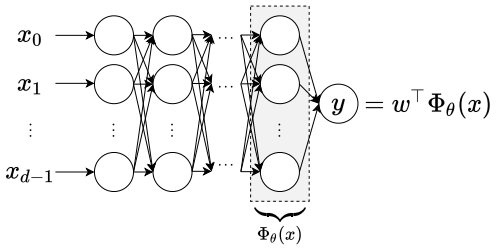

Figure 4. Training by optimizing with respect to all the weights in a neural network amounts to learning both the feature map and the associated weights in the output layer.

- **Ease of knowledge transfer:** One can do a so-called *transfer learning* [Bozinovski, 2020; Pan and Yang, 2009] with neural networks very easily by simply reusing weights from earlier and related neural network trainings. We used this technique extensively in our XVA applications in chapters 2 and 4, where the learning tasks associated with nearby time steps are necessarily close.

- **Approximating** $\operatorname{argmin} \mathbb{E}[.]$: The minimization of the expectation of a point-wise loss (in this example the squared distance between the output of the neural network and the variable to be projected) is done using the tools of stochastic programming [Shapiro et al., 2021] using a finite-sample approximation of the expectation. This approach is also known as *empirical risk minimization* in the context of stastistical learning [Vapnik, 1991]. In particular, we solve the minimization problem numerically by performing stochastic gradient descent (more precisely, using ADAM [Kingma and Ba, 2014]) on the empirical approximation[3]. Here we leverage the fact that we have access to the data generating process since the risk factors are driven by known stochastic differential equations (SDEs). The gradients with respect to the neural network parameters are computed exactly using *algorithmic differentiation* [Baydin et al., 2018; Savine, 2018], which is implemented by libraries such as JAX [Bradbury et al., 2018], PyTorch [Paszke et al., 2019] and Tensorflow [Abadi et al., 2016]. We refer to chapters 2 and 4 for detailed pseudo-codes of the training procedures that we used.

Assume now that one finished the approximation and is now given a candidate $h \in \mathcal{B}(\mathcal{X}, \mathbb{R})$. One way to measure whether this candidate is satisfactory is to see whether it is close enough (of course depending on some subjective threshold by the user of the approach) from the ground-truth value (*i.e.* $\mathbb{E}[Y|X]$) by computing the distance $\sqrt{\mathbb{E}[(h(X) - \mathbb{E}[Y|X])^2]}$. Unfortunately, the prospective user of our

3. This means that we do not draw completely new realizations at each SGD iteration. More precisely, we perform instead multiple *passes* or *epochs* over the data-set, where inside of each epoch we iterate over disjoint *mini-batches*. For each mini-batch we compute the empirical loss gradient by averaging over the mini-batch and perform a descent along the opposite direction of that gradient. The goal is to not have to spend an excessive amount of time on new simulations at every SGD step.

learning approaches usually has access only to couples of realizations (X, Y) and does not have direct access to $\mathbb{E}[Y|X]$, otherwise there would be no need to approximate it. Hence one can estimate $\mathbb{E}[(h(X)-Y)^2]$ directly but not, at least not via direct Monte Carlo, $\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2]$ and of course these two squared distances, while having the same argmin with respect to h on $\mathcal{B}(\mathcal{X}, \mathbb{R})$, are not necessarily interchangeable (see Figure 5).

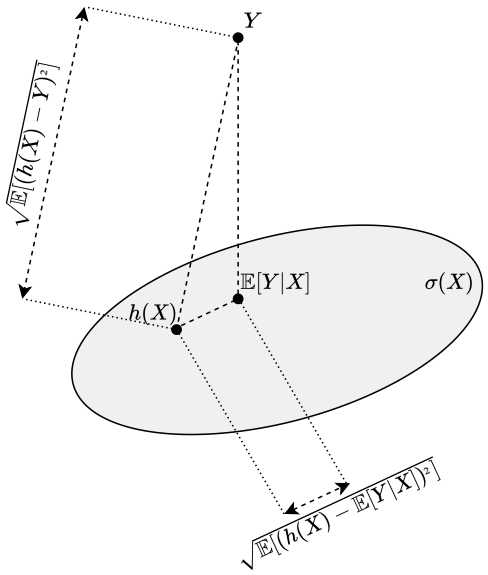

Figure 5. The distances $\sqrt{\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2]}$ and $\sqrt{\mathbb{E}[(h(X)-Y)^2]}$ are not interchangeable in general, unless Y is $\sigma(X)$-measurable.

One way to estimate this distance would be to approximate $\mathbb{E}[Y|X]$ using a nested Monte Carlo approach. However, this is slow and inefficient, and not suitable for live validation in a production setting. Instead, we developed a simple way in Chapter 2 to estimate our L^2 distance. In particular, we show that:

$$\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2] = \mathbb{E}[h(X)\,(h(X)-Y^{(1)}-Y^{(2)})+Y^{(1)}Y^{(2)}] \tag{4}$$

where $Y^{(1)}$ and $Y^{(2)}$ are two copies of Y that are independent conditional on X. Hence $h(X)\,(h(X)-Y^{(1)}-Y^{(2)})+Y^{(1)}Y^{(2)}$ is an unbiased estimator of our squared L^2 distance and it involves only two sub-simulations (as opposed to hundreds or even thousands if one were to approximate $\mathbb{E}[Y|X]$ directly using Nested Monte Carlo).

We extend the study of this learning approach to the approximation of the conditional Value-at-Risk and Expected-Shortfall in Chapter 3.

1.4 Non-Stationarity and Access to the Generating Process

We emphasize a key difference with respect to how Machine Learning is used in other disciplines. While in those disciplines, one usually has a finite data-set, with the data

coming either from a manual or automatic data collection and annotation, in the case of learning from simulated data we have by definition full access to the data generating process. In our case, the risk factors are usually modeled using SDEs or discrete Markov chains and hence our data-set is typically constructed via Monte Carlo simulations of paths of those stochastic processes and computing along each path the integrand that we wish to project using precise algorithmic steps. This makes issues like *over-fitting* less problematic as we can always generate more data if needed. Such issues are, however, theoretically still existent and one should still verify the results using out-of-sample metrics, computed using a sample that is independent of the training data and which is usually called a *test set*.

The conditional expectations or risk measures that we seek to approximate are needed at least on a daily basis. However, the model parameters (*e.g.* the parameters of the SDEs governing the risk factors) and other variables, such as the composition of the bank's portfolio in the case of our portfolio-wide XVAs, change continuously, hence introducing a certain non-stationarity. Thus, one cannot simply train the neural network approximators once and then use them in inference mode in every market condition at any subsequent date. Instead, one needs to regenerate the data-set by launching new Monte Carlo simulations under the new model parameters, and then retrain the neural networks using the new data-set and, in the case of XVAs, the bank's new portfolio.

In this setting, generating the data-set (*i.e.* Monte Carlo paths) and training are both part of the *final product*. This is precisely why we spent considerable efforts in devising simulations that are fast, using highly optimized CUDA kernels due to Monte Carlo simulations being intrinsically parallelizable, and training schemes which leverage many characteristics such as reusing weights through the time steps in BSDEs (chapters 2 and 4), the use of custom CUDA kernels to help with the generation of the labels for the training, and the extensive use of PyTorch's just-in-time (JIT) compilation mechanism. In Chapter 5, where we do not have this issue of non-stationarity but where fast training is still desirable in order to iterate faster in research and development, we provide an application where we went beyond these optimizations and implemented high performance training directly in `C++` using the `libtorch` library.

1.5 Related Generic Issues

There are a couple of questions that were voluntarily omitted during this thesis as we feel that these issues are already addressed or are being addressed by other researchers.

In particular, while we implemented our learning schemes using only one GPU, a successful large-scale implementation will necessarily need to use multiple GPUs and even multiple nodes.

On the simulation side, the different GPUs or nodes do not need to communicate between each other and they can perfectly generate their own Monte Carlo paths independently of each other by providing them with the simulation code but configuring the parallel random number generators in such a way that independence of the generated random numbers is guaranteed across GPUs and nodes [Abbas-Turki et al., 2014].

On the training side, the most challenging functional block is the SGD algorithm. One simple and naive way to parallelize it across multiple workers is to first synchronize them[4],

4. *i.e.* make sure they all have the same copies of neural network parameters.

then split the mini-batch[5] on which the average needs to be computed at a given SGD iteration across those workers and have each worker compute the values on the batch and average them, compute the associated gradients and then send the results back to a single *master* worker for final aggregation and leave all the other workers waiting while the master worker updates the neural network parameters using an SGD step. These synchronizations and locks create overhead which makes the parallelization across multiple workers less attractive. More recently, approaches that avoid locking and synchronization while having theoretical guarantees emerged and we cite in particular the HOGWILD! algorithm [Recht et al., 2011] which is natively supported by `PyTorch`. We refer to [Chen et al., 2016] for a detailed discussion of the merits and shortcomings of synchronous and asynchronous SGD in the context of distributed training.

Another issue is that of the optimization of hyper-parameters, *e.g.* the number of layers and neurons in a neural network. While we do not believe that this is something that needs to be done at each run, unless the model parameters and market conditions exhibit strong changes, we do believe that this is a fine-tuning that could be done occasionally as part of the maintenance of the neural network approximation codes. Exhaustive and naive approaches include *Grid Search*, where one tests all the combinations in a discrete grid of hyper-parameters and choses the ones that minimize the training objective (or another relevant cost function, depending on the task) evaluated using a data-set, usually called a *validation set*, that is independent from the one that was used for training but also independent from the test set. This approach, while very simple to implement, is slow and suffers from the curse of dimensionality because of the need to construct a grid of parameter values. More intelligent approaches are based on Bayesian optimization [Shahriari et al., 2015] and include bandit-based [Slivkins, 2019] algorithms such as HyperBand [Li et al., 2017], for which we refer to `Optuna` [Akiba et al., 2019] for a professional and stable implementation that is compatible with `PyTorch`.

Finally, we recognize that the idea of having to invest in infrastructures, in particular GPU clusters, in order to implement our proposed approaches in large-scale settings might seem intimidating. However, thanks to the availability of cloud solutions such as Amazon AWS or Google Cloud to name a few, one can *start small* with prototypes to assess both the applicability and the return-on-investment one can expect and then scale up as needed before investing in large in-house infrastructures.

2 Chapter Summaries

The mathematical notations in this section are local to each sub-section only.

2.1 Chapter 1 – XVA Analysis From the Balance Sheet

The 2008–2009 financial crisis reshaped the way in which derivatives are being priced.

5. No communication of data points is needed in our case as each worker already is supposed to have its own instance of the simulation code generating perfectly independent Monte Carlo paths.

That pushed banks to increasingly take into account valuation adjustments, called XVAs as introduced above, that make the task of pricing more nonlinear and necessitate a global approach to the pricing task, *i.e.* taking into account their entire portfolios. The capital structure model and valuation adjustment approach outlined in this chapter are rooted in a balance sheet and dividend policy perspective. This is aligned with shareholder interest and gives precise economic meaning to the different XVA terms.

In the proposed capital structure model, we mainly distinguish between *contra-asset (CA) desks*, which are in charge of counterparty risk and its funding implications, and *clean desks*, which are focused on the market risks related to their respective business lines. CA desks value contra-assets (*i.e.* CVA and FVA), charge them to clients in a trade-incremental way and make deposits in a reserve account which is then used for coping with the average losses due to counterparty risk and funding expenditures. Another important party in the capital structure model is the management which sources a risk margin, released in the form of KVA payments back to shareholders, as a risk premium on their *capital at risk*, *i.e* as the difference[6] between an Expected-Shortfall of the CA loss and the KVA, assumed to be loss-absorbing.

Assume[7] fully collateralized hedges, no recovery upon default, no variation margin on client deals and a stochastic risk-neutral pricing basis[8] $(\Omega, \mathcal{A}, \mathcal{F}, \mathbb{Q})$ with expectation operator $\mathbb{E}$ and $\mathbb{E}_t = \mathbb{E}[.|\mathcal{F}_t]$. In this simplified setup, the proposed capital structure model gives rise to the following XVA equations in continuous-time[9]:

$$
\begin{aligned}
\mathrm{CVA}_t &= \sum_{c\in\mathcal{C}} \mathbb{E}_t\left[\int_t^T \left(\mathrm{MtM}_s^{(c)}\right)^+ \delta_{\tau^{(c)}}(\mathrm{d}s)\right] \\
\mathrm{FVA}_t &= \mathbb{E}_t\left[\int_t^T \bar{\gamma}_s \left(\sum_{c\in\mathcal{C}} \mathrm{MtM}_s^{(c)} \mathbb{1}_{\{\tau^{(c)}>s\}} - \mathrm{CA}_s - \mathrm{CR}_s\right)^+ \mathrm{d}s\right] \\
\mathrm{CA}_t &= \mathrm{CVA}_t + \mathrm{FVA}_t \\
\mathrm{CR}_t &= \max\left(\mathrm{EC}_t, \mathrm{KVA}_t\right) \\
\mathrm{EC}_t &= \mathbb{ES}_t[L_{t+1} - L_t] \\
\mathrm{d}L_t &= \mathrm{dCA}_t + \underbrace{\sum_{c\in\mathcal{C}} \left(\mathrm{MtM}_t^{(c)}\right)^+ \delta_{\tau^{(c)}}(\mathrm{d}t)}_{\text{default losses}} + \underbrace{\gamma_t \left(\sum_{c\in\mathcal{C}} \left(\mathrm{MtM}_t^{(c)}\right)^+ \mathbb{1}_{\{\tau^{(c)}>t\}} - \mathrm{CA}_t - \mathrm{CR}_t\right)^+ \mathrm{d}t}_{\text{funding expenditures}} \\
\mathrm{KVA}_t &= \mathbb{E}_t\left[\int_t^T h\left(\mathrm{CR}_s - \mathrm{KVA}_s\right)^+ \mathrm{d}s\right]
\end{aligned}
$$

where T is the final maturity of the bank's portfolio (assumed to be held on a run-off basis), CVA_t, FVA_t, CA_t, CR_t, EC_t, KVA_t and L_t are the time t values of the respective CVA, FVA, contra-assets (*i.e.* CVA + FVA), capital-at-risk, economic capital, KVA and CA desk loss processes. $\mathcal{C}$ is a finite set indexing the counterparties of the bank, and for every $c \in \mathcal{C}$, $\mathrm{MtM}^{(c)}$ and $\tau^{(c)}$ are respectively the bank's mark-to-market process of the positions

6. More precisely the positive part of the difference, *i.e.* $(\mathrm{EC} - \mathrm{KVA})^+$ where EC is the Economic Capital, defined as a 97.5% Expected-Shortfall of the CA loss.

7. All these are however fully taken into account in the full chapter.

8. $\mathbb{Q}$ here is the bank survival measure.

9. These continuous-time equations are actually a particular case of Anticipated Backward Stochastic Differential Equations (ABSDEs) for which we give a general learning scheme in Chapter 4.

with counterparty c and its default time, assumed to be a stopping time with respect to $\mathcal{F}$. The process γ is the bank's funding spread and h is a constant *hurdle rate* representing the rate at which shareholders expect to be paid for their capital at risk.

$\mathbb{ES}_t$ is the conditional Expected-Shortfall operator defined for every $\mathcal{F}_T$-measurable random variable ℓ as follows:

$$\mathbb{ES}_t[\ell] = \mathbb{E}_t[\ell | \ell \geq \mathbb{V}a\mathbb{R}_t(\ell)]$$

and $\mathbb{V}a\mathbb{R}_t$ is the $\mathcal{F}$-conditional left-quantile of ℓ at the level α, also referred to as the Value-at-Risk at the confidence level α.

In a Markov setting, *i.e.* when $\mathbb{E}_t[.] = \mathbb{E}[.|\mathcal{X}_t]$ and $\mathbb{V}a\mathbb{R}_t$ coincides with the left-quantile conditional on $\mathcal{X}_t$ in the equations above for some risk-factors process $\mathcal{X}$, we propose to approximate the processes defined above using neural networks. For processes defined using conditional expectations, such as the CVA, FVA and the KVA above, we use least-squares regression against the risk factor process $\mathcal{X}$ using neural network approximators. For the conditional Expected-Shortfall, however, we do not have a direct *elicitability* result, *i.e.*, the functional representation of the conditional Expected-Shortfall is not a minimizer of a certain loss function[10]. However, from [Fissler and Ziegel, 2016; Fissler et al., 2016], the pair composed of the conditional Expected-Shortfall and the conditional quantile is *jointly elicitable*, *i.e.* one can recover both of them at the same time by minimizing a certain loss function. We thus implement a joint conditional Expected-Shortfall and Value-at-Risk learning algorithm which minimizes that loss function over a space of neural networks with outputs in $\mathbb{R}^2$ (*i.e.* outputting a couple of $\mathbb{ES}$ and $\mathbb{V}a\mathbb{R}$) as opposed to the usual scalar valued neural networks that we use in approximating conditional expectations.

We finally perform extensive numerical experiments showing that the proposed holistic approach can be implemented numerically and path-wise and trade-incremental XVAs can be computed efficiently using a combination of neural networks, Picard iterations and GPU computing.

2.2 Chapter 2 – Pathwise CVA Regressions With Oversimulated Defaults

In this chapter, we address the potential variance issues stemming from the need to specifically simulate defaults in our XVA approach in Chapter 1. In the case of the CVA, practitioners generally circumvent this issue by noticing that one can write it in *intensity-form* as follows:

$$\begin{aligned}\mathrm{CVA}_t &= \sum_{c\in\mathcal{C}} \mathbb{E}_t\Big[\int_t^T \big(\mathrm{MtM}_s^{(c)}\big)^+ \delta_{\tau^{(c)}}(\mathrm{d}s)\Big] \\ &= \sum_{c\in\mathcal{C}} \mathbb{E}_t\Big[\int_t^T \big(\mathrm{MtM}_s^{(c)}\big)^+ \gamma_s^{(c)} \exp\Big(-\int_t^s \gamma_u^{(c)}\,\mathrm{d}u\Big)\,\mathrm{d}s\Big]\end{aligned}$$

assuming that each counterparty $c \in \mathcal{C}$ has a stochastic default intensity process $\gamma^{(c)}$ such

10. unless one has access to the conditional quantile.

that, for every $0 \leq t < s$,

$$\mathbb{Q}(\tau^{(c)} > s | \mathcal{F}_t, \{\tau^{(c)} > t\}) = \mathbb{E}_t\left[\exp\left(-\int_t^s \gamma_u^{(c)} \, \mathrm{d}u\right)\right]$$

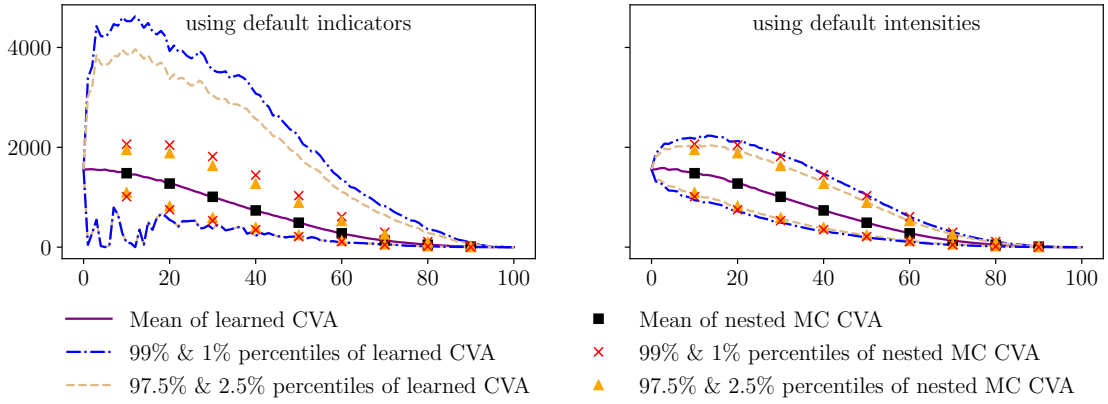

Figure 6. When the default indicators are absent from the integrand defining our CVA, regressions are more *well-behaved* as the variance of the integrand is reduced. ***x*-axis:** pricing time step, ***y*-axis:** level of the considered statistic of the CVA at the given time step.

However, this intensity-based work-around applies only to conditional expectations with integrands that are linear in the default indicators. In particular, this does not apply to the FVA, which has survival indicators inside a nonlinearity in the integrand, or the EC, whose definition depends on an expected shortfall of losses, including default losses and funding expenditures, which in turn depend on survival indicators.

To address this, we propose a simple simulation scheme where we separate the risk factors into two sub-groups: one group which we do not consider to be a significant contributor to our variance issue (in particular the diffusive factors) represented by a vector-valued process Y, and another composed of the major contributors to the variance (in our case the default indicators) and represented by another vector-valued process X. The main idea is then to simulate, at every time step, more realizations of X conditional on each realization of Y, given that simulating defaults conditional on diffusive risk factors is usually computationally inexpensive.

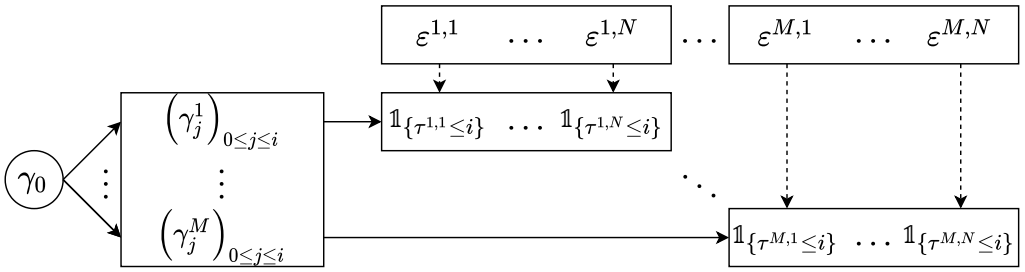

Figure 7. Proposed simulation scheme in the case of default events.

Assume for simplicity in this summary one single counterparty, which allows us to

skip the index of the counterparty in the intensity process notation γ, and assume all the considered processes are now in discrete time (*i.e.* γ_i is the discrete-time default intensity at the i-th time step of some time discretization). The idea in the case of our XVA setup is to first simulate a certain number of paths M of the diffusive risk factors (of which γ is part). Then, at every time step i, where we assume to have access to paths until i of γ, *i.e.* an i.i.d sample $\{(\gamma_j^k)_{0\leq j\leq i}\}_{1\leq k\leq M}$ of $(\gamma_j)_{0\leq j\leq i}$, we simulate conditional on each path, indexed by $k\in\{1,\dots,M\}$, N i.i.d realizations of default indicators, *i.e.* a sample $\{\mathbb{1}_{\{\tau^{k,l}\leq i\}}\}_{1\leq l\leq N}$ (using an i.i.d sample $\{\varepsilon^{k,l}\}_{1\leq l\leq N}$ of a standard exponential random variable).

This then allows one to define a sample of size $M\times N$ of any integrand in the XVA equations, or any other regression task that falls into our framework, by considering the different combinations of the conditioning index (k above) and the index of the *over-simulation* (l above). The resulting sample however does not consist of independent realizations. But we show numerically that such a sampling scheme is effective in tackling the variance issue described above for a low computational cost.

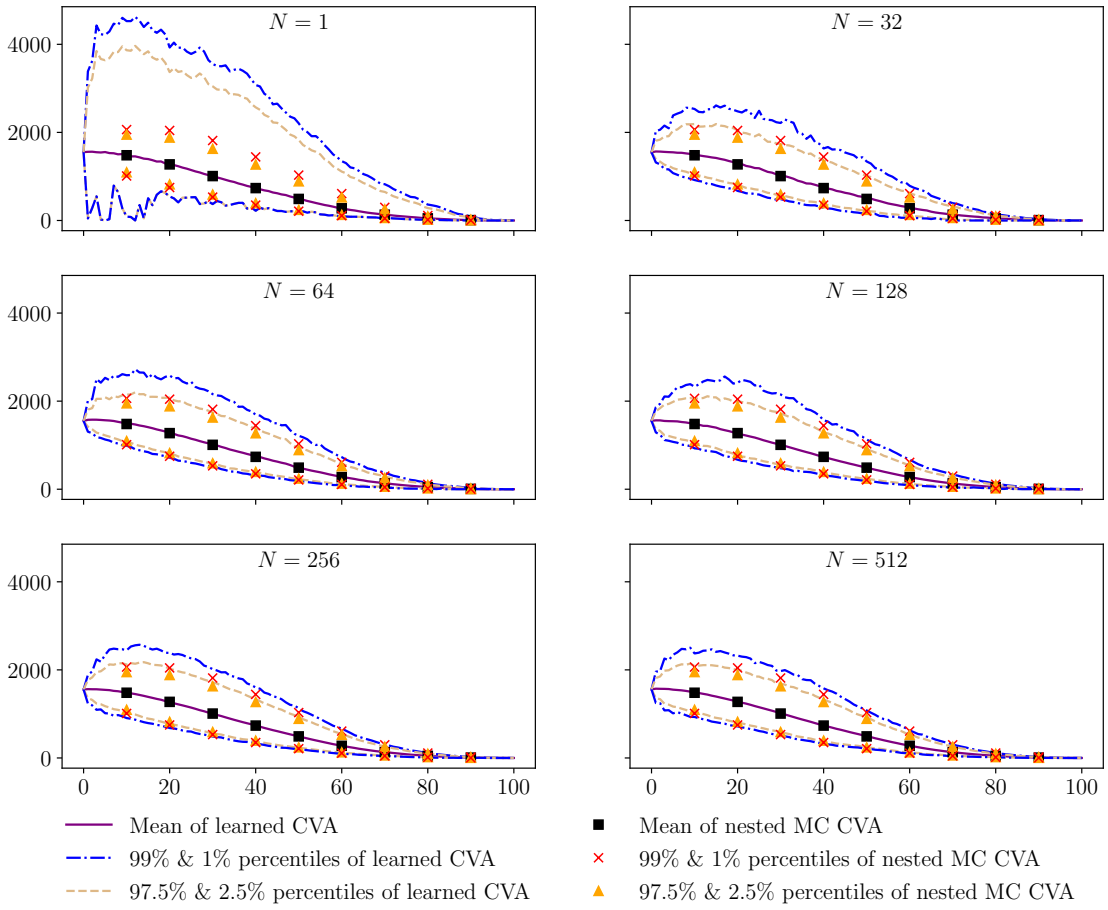

Figure 8. Starting from just $N=32$ more simulations of the defaults conditional on each realization of the diffusion risk factors (assuming a sample of size $M=2^{14}=16384$ of the diffusive realizations), we already get meaningful CVA profiles compared to the situation with no over-simulation of defaults (*i.e.* $N=1$). ***x*-axis:** pricing time step, ***y*-axis:** level of the considered statistic of the CVA at the given time step.

We also extend the *sample-average approximation* results of [Shapiro et al., 2021] to

our non i.i.d setup. We give statistical convergence guarantees in the form of a deviation inequality that helps drawing intuition as to how M and N impact the convergence of the minimum of the empirical loss (where the average is computed over our $M \times N$ realizations) to the minimum of the theoretical loss, *i.e.* where we use the expectation operator instead of an average over a finite sample.

Finally, in the case of conditional expectations, we address the problem of validating this learning approach in a live production setting without access to a nested Monte Carlo benchmark. We provide a procedure, introduced above in (4), to estimate the L^2 distance between the learned approximation and the ground-truth conditional expectation which does not require any knowledge of the latter.

2.3 Chapter 3 – Learning Value-at-Risk and Expected Shortfall

In this chapter, we study a two-step approach to learn the conditional Value-at-Risk and the conditional Expected Shortfall. Consider a probability space $(\Omega, \mathcal{A}, \mathbb{P})$ and let X be a random vector supported on a Polish space S and Y be an integrable scalar random variable. If we define the $\mathbb{V}a\mathbb{R}$ and $\mathbb{ES}$ as follows:

$$\begin{aligned} \mathbb{V}a\mathbb{R}(Y|X) &= \inf\{y \in \mathbb{R} : \mathbb{P}(Y \leq y|X) \geq \alpha\} \\ \mathbb{ES}(Y|X) &= \frac{1}{1-\alpha}\mathbb{E}[Y\,\mathbb{1}_{\{Y \geq \mathbb{V}a\mathbb{R}(Y|X)\}}|X] \end{aligned}$$

for some confidence level $\alpha \in (0,1)$, then there exist Borel measurable functions q and s such that

$$\begin{aligned} \mathbb{V}a\mathbb{R}(Y|X) &= q(X) \\ \mathbb{ES}(Y|X) &= s(X) \end{aligned}$$

Moreover,

$$\begin{aligned} q &\in \operatorname*{argmin}_{f \in \mathcal{B}^1(S)} \mathbb{E}\left[\frac{1}{1-\alpha}(Y - f(X))^+ + f(X)\right] \\ s &\in \operatorname*{argmin}_{f \in \mathcal{B}^2(S)} \mathbb{E}[(Y\,\mathbb{1}_{\{Y \geq q(X)\}} - f(X))^2] \end{aligned}$$

where $\mathcal{B}^1(S)$ and $\mathcal{B}^2(S)$ are the sets of Borel functions $f : S \to \mathbb{R}$ such that $f(X)$ is respectively integrable and square integrable. We give more general loss functions in Theorem 3.5 of this chapter. The conditional Value-at-Risk and the conditional Expected Shortfall can then be learned via the following *two-steps approach*:

1. Draw an i.i.d sample $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ of (X, Y);
2. Learn $\mathbb{V}a\mathbb{R}(Y|X)$ by finding $\hat{q} \in \operatorname{argmin}_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^n \frac{1}{1-\alpha}(Y_i - f(X_i))^+ + f(X_i)$;
3. Learn $\mathbb{ES}(Y|X)$ by finding $\hat{s} \in \operatorname{argmin}_{f \in \mathcal{G}} \frac{1}{n}\sum_{i=1}^n \Big(\hat{q}(X_i) + \frac{1}{1-\alpha}(Y_i - \hat{q}(X_i))^+ - f(X_i)\Big)^2$.

where $\mathcal{F}$ and $\mathcal{G}$ are two families of functions (also called the *hypothesis spaces*) over which we seek an approximation of respectively the Value-at-Risk and the Expected Shortfall. The second step approximates $\mathbb{ES}(Y|X) = \frac{1}{1-\alpha}\mathbb{E}[Y\,\mathbb{1}_{\{Y \geq \mathbb{V}a\mathbb{R}(Y|X)\}}|X]$ by replacing $\mathbb{V}a\mathbb{R}(Y|X)$ with the learned candidate $\hat{q}(X)$, *i.e.*

$$\begin{aligned}\mathbb{ES}(Y|X) &= \mathbb{V}a\mathbb{R}(Y|X) + \frac{1}{1-\alpha}\mathbb{E}[(Y - \mathbb{V}a\mathbb{R}(Y|X))^+|X] \\ &\approx \hat{q}(X) + \frac{1}{1-\alpha}\mathbb{E}[(Y - \hat{q}(X))^+|X].\end{aligned}$$

A non-asymptotic convergence analysis, rooted in results from the Rademacher and Vapnik-Chervonenkis theory [Shalev-Shwartz and Ben-David, 2014] and non-asymptotic bounds proven in [Barrera, 2022], is provided. Theorem 3.21 in particular states that:

$$\begin{aligned}c_{B_1}\mathbb{E}_n[(\hat{q}(X) - q(X))^2] \;\leq\; &\Big(2(2-\alpha)\inf_{f\in\mathcal{F}}\mathbb{E}[|f(X) - q(X)|]\Big) \wedge \Big(C_{B_1}\inf_{f\in\mathcal{F}}\mathbb{E}[(f(X) - q(X))^2]\Big) \\ &+ \frac{4(2-\alpha)B_2}{\sqrt{n}}\sqrt{2\log\left(\frac{2}{\delta}\right)} \\ &+ \frac{8(2-\alpha)B_1}{\sqrt{n}}\left(1 + \mathbb{E}\left[\sqrt{2\log\left(N_1\left(\mathcal{F}, X_{1:n}, \frac{B_1}{\sqrt{n}}\right)\right)}\right]\right)\end{aligned}$$

for every $\delta \in (0,1)$ with probability at least $1-\delta$, where $X_{1:n} = \{X_1, \ldots, X_n\}$, $\mathbb{E}_n[.] = \mathbb{E}[.|X_{1:n}]$ and we assume that $\mathcal{F}$ is uniformly bounded by $B_1 > 0$, with $\mathbb{V}a\mathbb{R}(Y|X)$ also assumed to be bounded by the same constant. We also assume that Y is bounded by a certain constant $B_2 > 0$ such that $B_1 \leq B_2$ and that there exist constants $C_{B_1} \geq c_{B_1} > 0$ such that

$$c_{B_1} \leq F'_{Y|X}(y) \leq C_{B_1}$$

$\mathbb{P}$-a.s for every $y \in [-B_1, B_1]$, where $F_{Y|X}$ is the cumulative distribution function of Y conditional on X. N_1 is a certain covering number function defined more in detail in Definition 3.17. A similar result is also shown for the approximation of the discrepancy between $\mathbb{ES}(Y|X)$ and $\mathbb{V}a\mathbb{R}(Y|X)$, *i.e* the difference $s - q$, in Theorem 3.25.

We also provide several learning schemes, using neural networks, to approximate both $\mathbb{V}a\mathbb{R}(Y|X)$ and $\mathbb{ES}(Y|X)$ for many confidence levels α at the same time. Among those schemes, we propose a novel one where we penalize the negative part of the derivative of the neural network approximator of the $\mathbb{V}a\mathbb{R}$ with respect to the confidence level α. α is randomized and considered as a covariate alongside X. We ensure numerically almost non-existent quantile crossing [Takeuchi et al., 2006; He, 1997; Koenker and Park, 1996].

In order to evaluate our proposed learning schemes numerically, we also performed two experiments: learning $\mathbb{V}a\mathbb{R}(Y|X)$ and $\mathbb{ES}(Y|X)$ in a Gaussian toy-model with first and second conditional moments of Y polynomial in X, and a more involved case-study where we learn a *Dynamic Initial Margin* in the same XVA calculation setting as in Chapter 2. For the purpose of the latter example, we provide a nested Monte Carlo procedure which performs a non-parametric learning $\mathbb{V}a\mathbb{R}(Y|X)$ using *conditional* stochastic gradient descents accelerated by initializing with a conditional Gaussian Value-at-Risk.

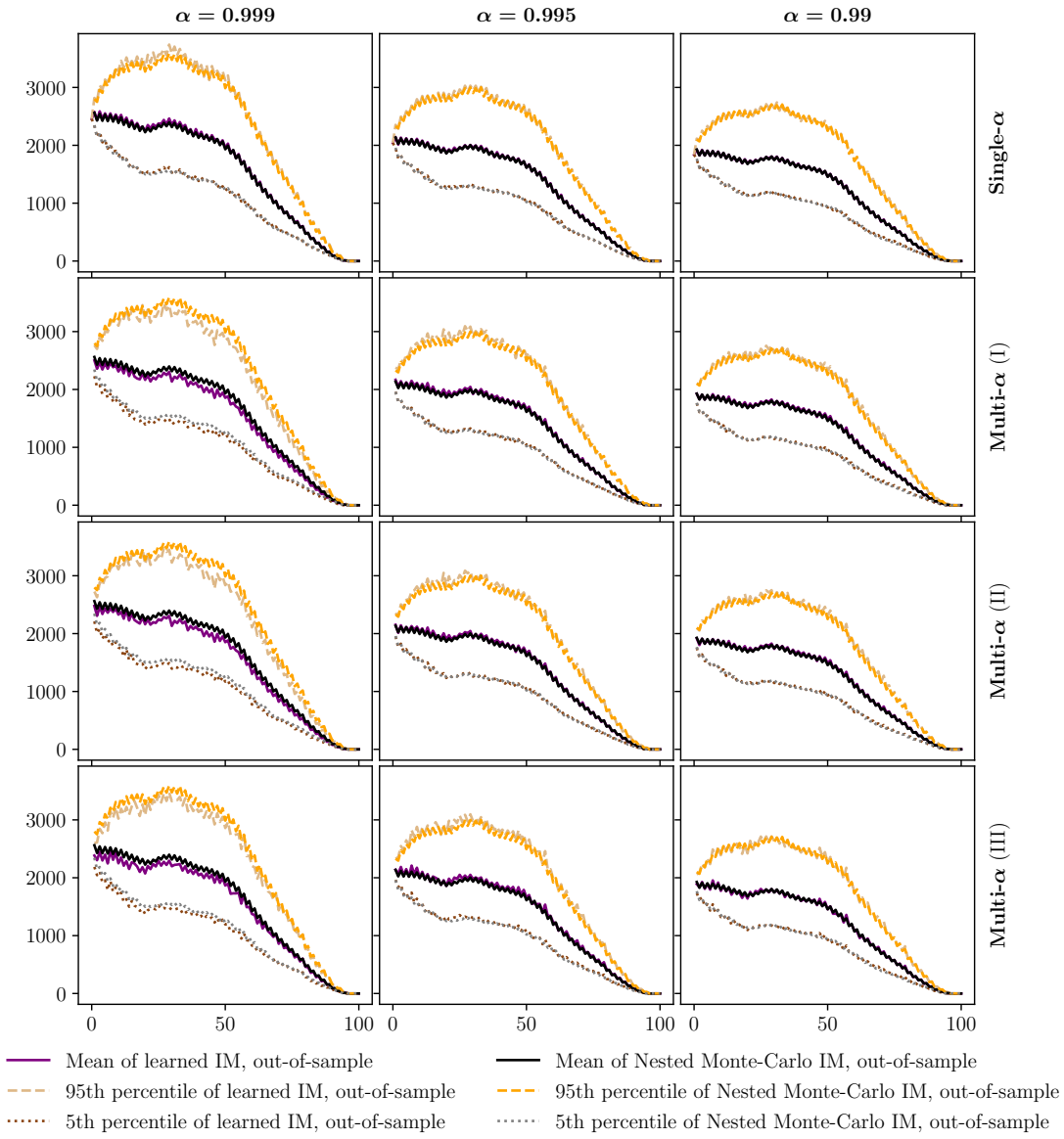

Figure 9. We successfully learn profiles of a dynamic initial margin (IM) for different confidence levels α in our XVA setting. Each column represents a given confidence level α and each row represents one of the learning schemes detailed in Chapter 3. ***x*-axis:** pricing time step, ***y*-axis:** level of the considered statistic of the IM at the given time step.

Finally, we also address the issue of validating this learning approach in settings where one does not have access to ground-truth values and cannot afford the computational cost of nested Monte Carlo benchmarks. We extend in particular the *twin simulation* trick introduced in Chapter 2 to the estimation of the distance in p-values (resp. in L^2) between the learned approximation and the ground-truth $\mathbb{V}a\mathbb{R}(Y|X)$ (resp. $\mathbb{ES}(Y|X)$).

2.4 Chapter 4 – Pathwise XVAs: The Direct Scheme

Assuming that the capital-at-risk is fungible for variation margin, we have seen in Chapter 1 that the FVA, at any time $0 \leq t \leq T$, can be written in continuous time as follows:

$$\mathrm{FVA}_t = \mathbb{E}_t\left[\int_t^T \bar{\gamma}_s \left(\sum_{c\in\mathcal{C}} \mathrm{MtM}_s^{(c)} \mathbb{1}_{\{\tau^{(c)}>s\}} - \mathrm{CA}_s - \mathrm{CR}_s\right)^+ \mathrm{d}s\right]. \tag{5}$$

However, we have $\mathrm{CA} = \mathrm{CVA} + \mathrm{FVA}$ and the capital-at-risk term CR depends on the economic capital EC which is in turn defined as an Expected Shortfall of the contra-assets desk's future losses. These involve losses due to the variation of the FVA and funding expenditures. Equation (5) belongs to the class of *anticipated backward stochastic differential equation* (ABSDE) [Peng and Yang, 2009]. Crépey et al., 2020 show the existence of a unique solution to such ABSDEs when the anticipative term in the *driver* depends on a conditional Expected Shortfall of future increments of the martingale part of the solution. In this chapter, we are interested in solving numerically ABSDEs of the form:

$$Y_t = \mathbb{E}_t\left[\phi(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, \mathbb{ES}_s(\Phi_{\bar{s}}(M)))\,\mathrm{d}s\right] \tag{6}$$

where Y is a special semimartingale with canonical Doob-Meyer local martingale component M, $\mathcal{X} = (X, J)$ with X an $\mathbb{R}^p$-valued strong solution of an SDE and J a $\{0,1\}^q$-valued *Markov chain like* component, ϕ is a continuous function from $\mathbb{R}^p$ to $\mathbb{R}^l$, f is a continuous function from $[0,T] \times \mathbb{R}^p \times \mathbb{R}^l \times \mathbb{R}$ to $\mathbb{R}^l$ satisfying certain integrability and Lipschitz regularity assumptions, $\bar{\cdot}$ is a deterministic operator which maps each time t into a time $\bar{t} \in [t, T]$, $\mathbb{ES}_s$ is a conditional Expected Shortfall as previously defined in Chapter 1, and $\Phi_{\bar{s}}$ is defined for every $s \in [0,T]$ as follows:

$$\Phi_{\bar{s}}(M) = \Phi(s; \mathcal{X}_{[s,\bar{s}]}, M_{[s,\bar{s}]} - M_s)$$

where $\Phi(s; \mathbf{x}, \mathbf{m})$ is a real valued deterministic map of time s and càdlàg paths $\mathbf{x}$ and $\mathbf{m}$ on $[s, \bar{s}]$ such that $\mathbf{m}_s = 0$, and M is the martingale part of Y.

While we solve a particular case of ABSDEs in Chapter 1 using a learning approach combined with Picard iterations, we propose in this chapter a one-shot scheme that does not require Picard iterations, to solve more general ABSDEs in the form of (6). More precisely, we propose the following explicit discrete time scheme:

$$\begin{aligned} Y^h_{t_i} &= \mathbb{E}_{t_i}[Y^h_{t_{i+1}} + f(t_i, \mathcal{X}^h_{t_i}, Y^h_{t_{i+1}}, \rho^h_{t_{i+1}})\,\Delta t_{i+1}] \\ \rho^h_{t_i} &= \mathbb{ES}_{t_i}\left(\Phi^h_{t_i}\left(Y^h_{t_\ell} + \sum_{k<\ell} f(t_k, \mathcal{X}^h_{t_k}, Y^h_{t_{k+1}}, \rho^h_{t_{k+1}})\,\Delta t_{k+1}, \ell = 0, \ldots, n\right)\right) \end{aligned}$$

where we consider a time-grid $0 = t_0 < t_1 < \cdots < t_n = T$, $\Delta t_{i+1} = t_{i+1} - t_i$, $\bar{t_i}$ is assumed in this context to be approximated on this time grid, $\mathcal{X}^h$ is a simulatable approximation of $\mathcal{X}$ in discrete time (*e.g.* using an Euler scheme for X) and $\Phi^h_{\bar{t_i}}$ is a certain computable discrete time approximation of $\Phi_{\bar{t_i}}$. This proposed scheme is then implemented using least-squares (see Chapter 2) and quantile (see Chapter 3) regressions using neural network approximators. In the absence of a nested Monte Carlo benchmark, we estimate local L^2 regression errors using the twin simulation approach introduced in Chapter 2. In addition to the simplicity in the formulation of our explicit scheme, we demonstrate on an XVA case-study the superiority, both in terms of computational speed and stability with respect to the size of the time steps, of our approach compared to schemes based on Picard iterations.

2.5 Chapter 5 – Fast Calibration using Complex-Step Sobolev Training

Given a parametrized pricing model, its calibration usually seeks to solve a minimization

problem of the form:

$$\theta^\star \in \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_{l=1}^{L} \left(p_{\text{model}}(\theta, k^{(l)}, \tau^{(l)}) - p^{(l)}_{\text{mkt}}\right)^2$$

where the calibration instruments are vanilla European calls, $p_{\text{model}}(\theta, k, \tau)$ is the price of a vanilla call with strike k and time-to-maturity τ in our pricing model assuming parameters $\theta \in \mathbb{R}^n$, Θ is a set of admissible model parameters, and we have access to L market prices of vanilla calls $p^{(1)}_{\text{mkt}},...,p^{(L)}_{\text{mkt}}$ corresponding to strikes $k^{(1)},...,k^{(L)}$ and maturities $\tau^{(1)},...,\tau^{(L)}$.

Unless in special settings where the minimization above can be solved in closed form (*e.g.* in a local volatility model using Dupire's formula), one generally has to solve the problem by using a numerical optimization routine which usually involves repeated evaluations of the objective function (and its gradient if using a gradient-based optimizer). The computational cost of the calibration can sometimes be prohibitive if p_{model} is difficult to compute, *i.e.* if one does not have an analytic or semi-analytic formula for the model price, as is the case for example for rough volatility models [Bayer et al., 2016].

To circumvent this issue, approaches have been proposed where one first constructs a fast approximation of p_{model} as a function of model and product parameters using Machine Learning techniques, *e.g.* based on neural networks [Horvath et al., 2021; Bayer and Stemper, 2018] or gaussian process regressions [De Spiegeleer et al., 2018]. These approaches learn an approximation of the model pricing function using pre-constructed data-sets consisting of combinations of model/product parameters and model prices obtained using a Monte Carlo routine, hence effectively treating the model pricing function as a black-box.

In this chapter, we propose to leverage the fact that the model price is a conditional expectation of the considered payoff, which we may denote Z^Ξ, *i.e.*

$$p_{\text{model}}(\Xi, K, T) = \mathbb{E}[Z^\Xi | \Xi, K, T]$$

where Ξ, K and T are randomized versions, in a sense to be specified, of respectively the model parameters vector, the strike and the maturity. We neglect the discounting for the sake of this introduction. Hence, the model price corresponding to model and product parameters Ξ, K and T, when seen as an L^2 projection of Z^Ξ on the vector subspace consisting of square integrable $\sigma(\Xi, K, T)$-measurable random variables, is a minimizer of the associated L^2 projection error, *i.e.*

$$p_{\text{model}} \in \underset{\varphi\in\mathcal{B}}{\operatorname{argmin}}\, \mathbb{E}[(\varphi(\Xi, K, T) - Z^\Xi)^2].$$

One could then carry out the minimization over a suitable space of neural networks $\mathcal{N}$ as we did in the previous chapters for similar projection problems, *i.e.* seek an approximation p_{proxy} such that

$$p_{\text{proxy}} \in \underset{\varphi\in\mathcal{N}}{\operatorname{argmin}}\, \mathbb{E}[(\varphi(\Xi, K, T) - Z^\Xi)^2].$$

In the present chapter[11], however, we propose to augment the learning using *path-wise derivatives* of the payoff with respect to model and product parameters, *i.e.* we instead

11. We did not use this approach, at least in its current form, in the XVA setting of the previous chapters because it is challenging from the point of view of memory space occupation, as path-wise derivatives will have to be computed and kept in memory at every coarse time step. Moreover, this approach does not apply directly to conditional risk-measures, as one cannot for example interchange conditional Value-at-Risk and differentiation operators.

solve the following minimization problem

$$p_{\text{proxy}} \in \underset{\varphi \in \mathcal{N}}{\operatorname{argmin}}\, \mathbb{E}[(\varphi(\Xi, K, T) - Z^{\Xi})^2] + \sum_{k=1}^{n+2} \lambda_k \mathbb{E}[(\partial_k \varphi(\Xi, K, T) - \partial_k Z^{\Xi})^2] \tag{7}$$

where $\lambda \in (\mathbb{R}_+^\star)^{n+2}$, the networks in $\mathcal{N}$ are assumed to be sufficiently regular and ∂_k is the partial derivative with respect to the k-th component of the concatenation of Ξ and (K, T), with $\partial_k Z^{\Xi}$ a so-called *path-wise derivative* [Broadie and Glasserman, 1996] which we recall and for which we give sufficient assumptions such that one has

$$\partial_k \mathbb{E}[Z^{\Xi}|\Xi, K, T] = \mathbb{E}[\partial_k Z^{\Xi}|\Xi, K, T]. \tag{8}$$

Thus, in light of (8), the learning problem expressed in (7) then explicitly seeks to project both the payoff and its path-wise derivatives. Augmenting the learning with information on the derivatives was first[12] studied in a neural network context in [Czarnecki et al., 2017] under the name of *Sobolev training*, with Huge and Savine, 2020 later applying the same approach to the learning of prices seen as functions of the initial values of the SDE of the underlying. In an empirical risk minimization setting, this approach is more efficient than brute-force simulation of more paths, as path-wise derivatives can share not only the same random numbers but also usually many common sub-expressions.

We also show that one can avoid the computational burden associated with the evaluation of the squared errors for each partial derivative in (7) and that one can instead compute a squared projection error associated with only one directional derivative in a randomized direction. Indeed, we have:

$$\mathbb{E}[(u^\top \nabla \varphi(\Xi, K, T) - u^\top \nabla (e^{-rT} Z^{\Xi}))^2] = \sum_{k=1}^{n+2} \lambda_k \mathbb{E}[(\partial_k \varphi(\Xi, K, T) - \partial_k (e^{-rT} Z^{\Xi}))^2]$$

for any L^2-integrable random vector u supported on $\mathbb{R}^{n+2}$ with zero-mean components such that $\operatorname{cov}(u) = \operatorname{diag}(\lambda_1, \ldots, \lambda_{n+2})$ and u is independent of Ξ, K, T, Z^{Ξ}. We also show how to choose the distribution over u such that the added[13] variance in the integrand in (7) is minimized.

We then propose to compute the directional derivative $u^\top \nabla \varphi(\Xi, K, T)$ with an error down to machine precision using *complex-step differentiation* [Martins et al., 2003; Squire and Trapp, 1998] while remaining in single precision by introducing a small imaginary perturbation in the direction of differentiation, *i.e.*

$$\frac{1}{\varepsilon} \operatorname{Im}(\varphi(\Xi + \mathrm{i}\,\varepsilon\, u_{1:n}, K + \mathrm{i}\,\varepsilon\, u_{n+1}, T + \mathrm{i}\,\varepsilon\, u_{n+2})) = u^\top \nabla \varphi(\Xi, K, T) + \mathcal{O}(\varepsilon^2)$$

as $\varepsilon \to 0$ whenever φ is analytic and at least thrice differentiable with respect to its inputs and where i is the imaginary unit. In contrast, the finite difference method usually fails for step sizes that are too small because of round-off errors unless one moves to double precision, which would harm performance on the GPU and would use twice the amount of memory that is used in single precision.

12. At least to our knowledge.

13. Indeed, as we show in the chapter, randomizing the direction of differentiation necessarily increases the variance of the integrand in (7).

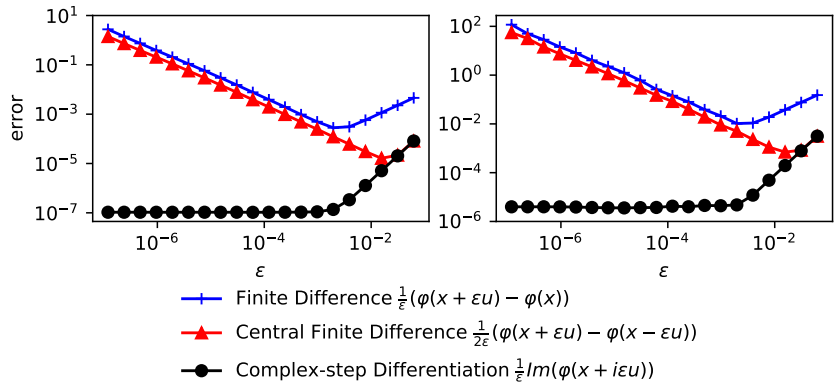

Figure 10. Sample average of the absolute value of (**left**) absolute and (**right**) relative errors, when approximating the directional derivative of a randomly initialized neural network φ, with 28 inputs, 6 hidden layers, 112 hidden units per layer and an analytic Softplus activation, with respect to its inputs, using each of the finite difference, central finite difference and complex-step differentiation methods. The average is done over an i.i.d sample of $2^{14} = 16384$ errors each corresponding to a network input vector x with components drawn independently from $\mathcal{U}([-\sqrt{3}, \sqrt{3}])$ and a direction u drawn as in Proposition 5.10 with $\lambda_1 = \cdots = \lambda_{28} = 1$. Plots are in log-log scale.

We show that this approach to computing the randomized directional derivatives is faster than computing them exactly using vector-jacobian products in forward mode. We dub the resulting learning approach *complex-step sobolev training*.

We show the effectiveness of our method by testing it numerically on a 5×5 fixed-grid local volatility example model where each local volatility node is treated as a model parameter, *i.e.* yielding 25 pricing model parameters in total. We also provide benchmarks showing the speedups gained using our proposed approach and show that it is statistically more efficient than simply brute-force generating more Monte Carlo paths.

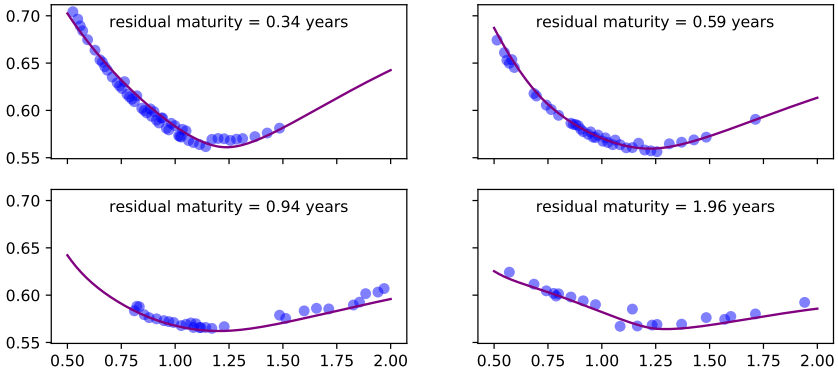

Figure 11. Fit of TSLA implied volatility smiles on 2022/02/14 using a 5×5 local volatility model calibrated using our proxy pricer. **Blue dots:** market prices, **purple curve:** implied volatility smile of fitted local volatility model, ***x*-axis:** moneyness, ***y*-axis:** implied volatility levels.

Introduction (français)

1 Contexte

1.1 L'ère du Big Data et de l'apprentissage automatique

Le 21ème siècle est marqué par une abondance de données [Sirko et al., 2021; Sidorov et al., 2020; Kuznetsova et al., 2020; Abu-El-Haija et al., 2016; Deng et al., 2009], l'émergence de nouvelles technologies pour les structurer et les gérer [Zaharia et al., 2016; Lakshman and Malik, 2010; Borthakur, 2007], et des innovations matérielles continues pour les traiter dans des délais de plus en plus courts [Nickolls and Dally, 2010]. Ceci a défini l'ère dite du *"Big Data"* [Magoulas and Lorica, 2009], marquée par l'omniprésence et la popularité des techniques et des architectures de programmation parallèle, l'utilisation généralisée d'architectures de calcul massivement parallèles telles que les unités de traitement graphique (GPUs) via des APIs de programmation comme CUDA [Luebke, 2008] et OpenCL [Stone et al., 2010] en raison de la dépendance de la plupart des algorithmes de pointe des routines d'algèbre linéaire qui bénéficient grandement de l'accélération fournie par les GPUs [Shi et al., 2016], et la prévalence des algorithmes d'optimisation stochastique, de la version originelle de la descente du gradient stochastique (SGD) étudiée dans les travaux fondateurs de Robbins and Monro, 1951 à des variantes plus récentes et qui ont connu beaucoup de succés telles que l'algorithme ADAM de Kingma and Ba, 2014, grâce aux propriétés de passage à l'échelle attrayantes de ces algorithmes [Bottou, 2010].

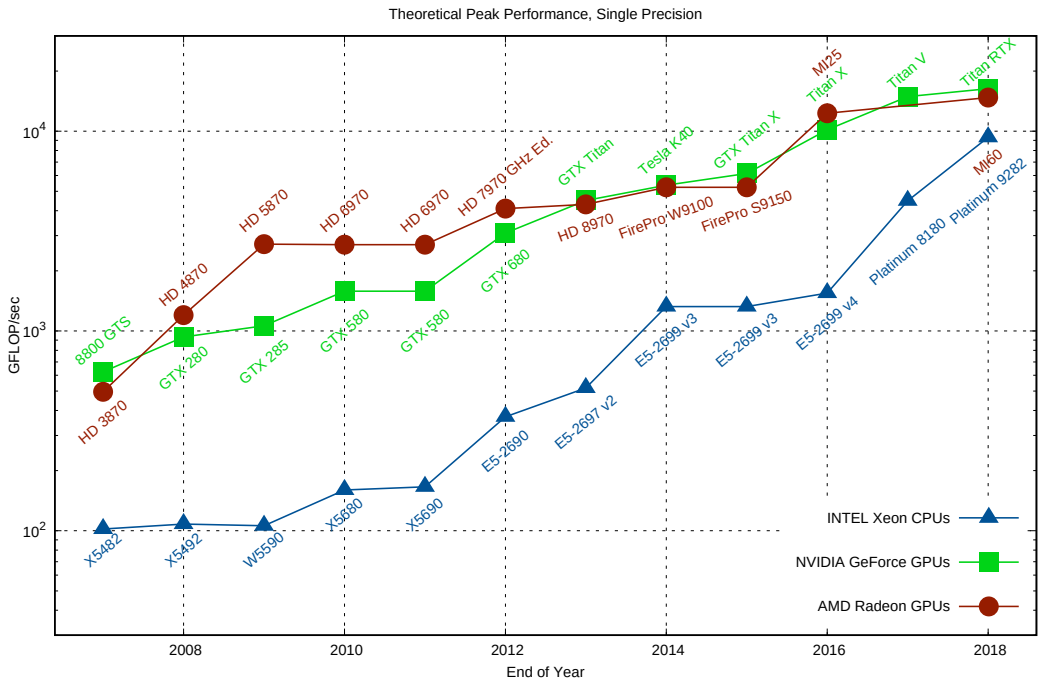

Figure 1. Pic théorique d'opérations en virgule flottante par seconde (FLOP/s, 1 GFLOP/s=10^9 FLOP/s) pour des CPUs Intel vs des GPUs AMD et NVidia en simple précision. Produite en utilisant un code et des données par Karl Rupp disponibles sur https://github.com/karlrupp/cpu-gpu-mic-comparison sous licence CC BY 4.0.

Avec cette révolution dans les architectures de calcul, la programmation parallèle, l'optimisation stochastique et autres méthodes numériques passant à l'échelle, un autre paradigme de conception de programmes informatiques a pris de l'importance. Dans ce nouveau paradigme, les programmeurs n'ont pas à écrire explicitement la solution à un problème. En utilisant des techniques d'apprentissage statistique, ils peuvent, à la place, implémenter un algorithme qui apprend la solution, ou du moins à s'en approcher, à condition d'avoir des quantités suffisantes de données. Ce paradigme est plus pertinent que jamais à l'ère du Big Data. L'essence de celui-ci peut être mieux résumée par les citations suivantes de Samuel, 1959, qui a inventé le terme Machine Learning, ou *apprentissage automatique*, dans le contexte du jeu de dames:

> [...] *un ordinateur peut être programmé pour qu'il apprenne à jouer au jeu de dames mieux que la personne qui a écrit le programme. De plus, il peut apprendre à le faire dans un laps de temps remarquablement court.* [...] *La programmation d'ordinateurs pour apprendre à partir de l'expérience devrait éventuellement éliminer le besoin d'une grande partie de cet effort de programmation détaillée.*

Les approches d'apprentissage automatique, et plus particulièrement celles basées sur les réseaux de neurones, ont connu un énorme succès dans des disciplines telles que la vision par ordinateur [Janai et al., 2020; Voulodimos et al., 2018], le traitement du langage naturel [Wolf et al., 2019; Young et al., 2018], les systèmes de recommandation [Zhang et al., 2019; Adomavicius and Tuzhilin, 2005] ou encore la détection d'anomalies [Chalapathy and Chawla, 2019; Zenati et al., 2018; Omar et al., 2013].

Bien qu'il y ait eu une certaine adoption par les institutions financières, principalement dans le secteur de la banque de détail avec des applications telles que le *credit scoring* [Lessmann et al., 2015] ou la détection de fraude [Chan et al., 1999], son usage est resté limité en finance quantitative, notamment du côté de la vente. Cela est, à notre avis, principalement dû à une idée fausse selon laquelle l'apprentissage automatique ne s'appliquerait qu'aux problèmes avec des applications réelles et des données empiriques, par opposition aux modèles mathématiques et au cadre de l'évaluation risque-neutre dans les problèmes d'évaluation et de couverture de produits dérivés. Des applications ont néanmoins émergé, notamment celles de l'évaluation rapide (*fast pricing*) [Horvath et al., 2021; De Spiegeleer et al., 2018] où l'objectif est de construire des approximations de prix rapides qui remplaceraient des routines d'évaluation lentes, comme celles basées sur des simulations de Monte Carlo. Goudenege et al., 2020 proposent de combiner une méthode d'arbre en une étape [Ekvall, 1996] avec une régression par processus gaussiens [Rasmussen and Williams, 2006] afin d'approximer la valeur de continuation aux dates d'exercice lors de la valorisation d'une option bermudienne via une programmation dynamique rétrograde. Ces applications sont sans aucun doute essentielles car elles constituent des éléments de base importants desquels dépendent fortement les transactions, les calculs réglementaires et de risques, la calibration de modèles et le trading électronique, entre autres, à la fois sur le plan fonctionnel et des performances. En effet, des bibliothèques d'évaluation plus rapides conduisent nécessairement à des calculs de risque plus rapides. Dans ce cadre, les algorithmes d'apprentissage automatique sont utilisés dans un but d'ajustement de courbes.

De nouveaux usages de l'apprentissage automatique spécifiques aux modèles mathématiques de la finance quantitative ont commencé à émerger grâce à des travaux fondateurs tels que [Huré et al., 2020; Raissi, 2018; Han et al., 2018]. En reformulant des *équations différentielles stochastiques rétrogrades* (BSDEs) [Pardoux and Peng, 1990] comme des

problèmes de contrôle stochastique en temps discret, les auteurs arrivent à résoudre des BSDEs ou des *équations aux dérivées partielles* (PDEs) en utilisant un espace de réseaux de neurones avec une architecture appropriée comme espace de recherche dans le problème de minimisation ainsi que des trajectoires simulées via Monte Carlo comme leur jeu de données d'*entraînement*. L'applicabilité des BSDEs en finance a été mise en évidence pour la première fois dans [El Karoui et al., 1997] et un traitement plus récent est fait dans [Crépey, 2013]. Buehler et al., 2019 utilisent une approche formellement similaire pour aborder directement le problème de la couverture d'un produit dérivé en présence d'imperfections de marché comme la présence des coûts de transaction, des effets de glissement (*slippage* en anglais) et l'impact de marché, souvent négligés dans le cadre usuel de l'évaluation risque-neutre. Becker et al., 2019 proposent également une nouvelle technique pour résoudre les problèmes d'arrêt optimal en représentant la politique (*policy* en anglais) à l'aide de réseaux de neurones et montrent des applications dans la valorisation de dérivés bermudiens ou avec clauses de rappel. Cela a marqué le début d'une tendance où l'apprentissage automatique dans la finance va au-delà de l'ajustement de courbes et exploite le cadre mathématique propre aux modèles utilisés par les banques.

Cette thèse s'inscrit également dans la même ligne de pensée en proposant des schémas basés sur l'apprentissage automatique, ou plus précisément des schémas basés sur des réseaux de neurones, pour résoudre les problèmes de calcul d'*ajustements de valeurs*, *X-Valuation Adjustments* (XVAs) en anglais, et des mesures de risque conditionnelles et l'accélération de calibration de modèles de valorisation.

1.2 Un bref intermezzo sur les XVAs

La crise financière de 2008–2009 a vu de nombreuses réformes bancaires visant à renforcer la robustesse du système financier. Une conséquence a été l'utilisation accrue des métriques XVA lors des valorisations des produits dérivés. Ces métriques ont pour but d'aider à quantifier et évaluer l'incomplétude du marché par les banques et à tenir compte du risque de contrepartie et de ses répercussions en termes de capital et de financement. La lettre X est à remplacer par:

- C pour *credit*, *i.e.* Credit Valuation Adjustment (CVA);
- D pour *debt*, *i.e.* Debt Valuation Adjustment (DVA);
- F pour *funding*, *i.e.* Funding Valuation Adjustment (FVA);
- M pour *margin*, *i.e.* Margin Valuation Adjustment (MVA);
- K pour *capital*[1], *i.e.* Capital Valuation Adjustment (KVA).

Les applications XVA constituent une partie importante de cette thèse en raison de leur importance dans les calculs réglementaires. Ceci est fait dans le cadre défini dans [Albanese et al., 2021] (travail qui est également présenté dans le Chapitre 1), abordant des caractéristiques raffinées telles que la simulation de défauts (voir Chapitre 2) ou la fongibilité du capital de réserve et du capital à risque avec la marge de variation (voir Chapitre 4).

Pour les besoins de cette introduction, nous donnons brièvement un exemple avec la CVA et la FVA pour mettre en évidence la complexité du calcul de ces métriques. En particulier, nous abordons le problème du calcul de ces métriques à des pas de temps futurs dans le cadre d'un modèle risque-neutre sur les facteurs de risque de marché et de crédit.

1. la lettre C est déjà prise pour la CVA.

Réussir à modéliser l'évolution future de la CVA, et en particulier la modéliser comme un processus stochastique, est essentiel car des pertes peuvent se matérialiser du seul fait de la fluctuation de cette métrique. Ainsi par exemple, le Comité de Bâle a déclaré ce qui suit dans [The Bank for International Settlements, 2011]:

> *Pendant la crise financière, environ deux tiers des pertes attribuées au risque de contrepartie étaient dues à des pertes de CVA et seulement un tiers environ était dû à des événements de défaut.*

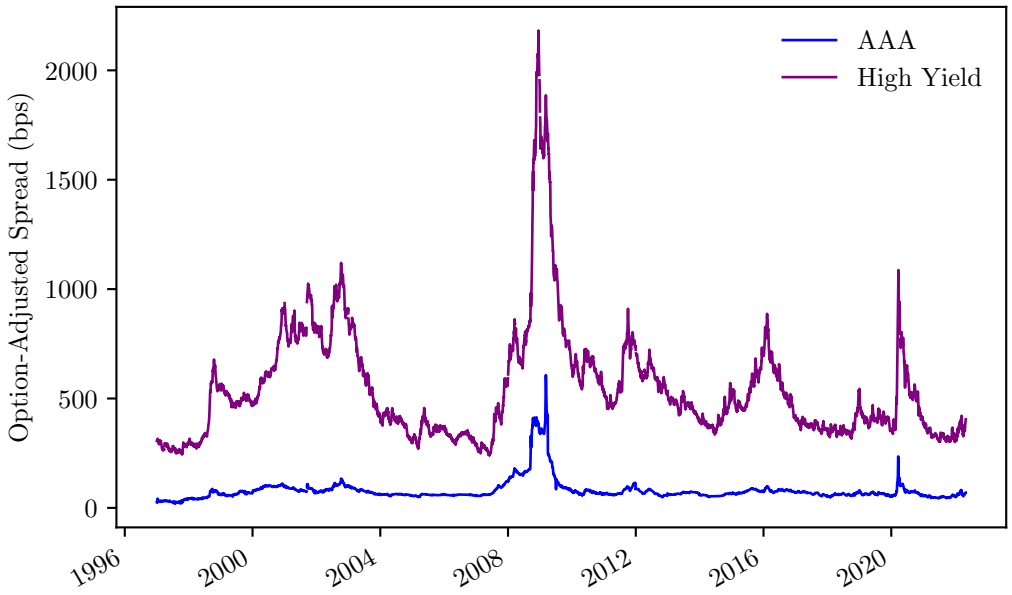

Figure 2. Les pertes de CVA peuvent être dues simplement par des fluctuations de sa valeur causées par des changements dans les *spreads* de crédit. Ce graphe représente les *Option-Adjusted Spreads* des indices ICE BofA US High Yield Index **(violet)** and the ICE BofA AAA US Corporate Index **(bleu)**. Données récupérées depuis FRED, Federal Reserve Bank of St. Louis; https://fred.stlouisfed.org/series/BAMLH0A0HYM2 et https://fred.stlouisfed.org/series/BAMLC0A1CAAA.

En supposant pour simplifier une seule contrepartie et une seule transaction non-collatéralisée conclue par la banque et le client sur un produit dérivé, un recouvrement nul en cas de défaut, en négligeant l'actualisation, et en supposant une discrétisation temporelle $0 = t_0 < t_1 < \cdots < t_i < \cdots < t_n = T$ où T est la maturité du produit, on définit le processus CVA en temps discret conme suit:

$$\mathrm{CVA}_i := \mathbb{E}\left[\sum_{j=i}^{n-1} \mathrm{MtM}_{j+1}^{+} \mathbb{1}_{\{t_j < \tau \leq t_{j+1}\}} \middle| X_i, \mathbb{1}_{\{\tau > t_i\}} \right] \tag{1}$$

pour tout $i \in \{0, \ldots, n-1\}$, où MtM_j est la valorisation, ou *mark-to-market*, du produit du point de vue de la banque (*i.e.* positif lorsqu'il s'agit d'un actif pour la banque et négatif lorsqu'il s'agit d'un passif) à l'instant t_j, τ est le temps de défaut de la contrepartie, X_i est un vecteur de facteurs de risque de marché à l'instant t_i et on suppose un certain modèle de valorisation multi-facteurs dans une base stochastique de valorisation. Définie de cette manière, la CVA est une anticipation des pertes futures dues aux défauts, conditionnellement à l'état du marché (via X) et des contreparties (via $\mathbb{1}_{\{\tau > .\}}$). Une méthode *brute-*

force pour simuler des réalisations de l'espérance conditionelle en (1), par exemple pour les injecter ensuite dans d'autres non-linéarités ou XVAs d'ordre supérieur comme la FVA ou simplement pour calculer une Value-at-Risk dessus, consisterait à utiliser un estimateur tel que:

$$\widehat{\mathrm{CVA}}_i^{(k)} := \frac{1}{M}\sum_{l=1}^{M}\sum_{j=i}^{n-1}\left(\mathrm{MtM}_{j+1}^{(k,l)}\right)^+ \mathbb{1}_{\{t_j<\tau^{(k,l)}\le t_{j+1}\}}$$

pour $k\in\{1,...,N\}$ où on dispose d'un échantillon i.i.d $\left\{\left(X_i^{(k)},\tau^{(k)}\right)\right\}_{1\le k\le N}$ de (X_i,τ) et pour chaque $k\in\{1,\dots,N\}$ d'un échantillon indépendant $\left\{\left(\mathrm{MtM}_{i+1}^{(k,l)},\dots,\mathrm{MtM}_n^{(k,l)},\tau^{(k,l)}\right)\right\}_{1\le l\le M}$ de $(\mathrm{MtM}_{i+1},\dots,\mathrm{MtM}_n,\tau)$ conditionnellement à $(X_i,\mathbb{1}_{\{\tau>t_i\}})=\left(X_i^{(k)},\mathbb{1}_{\{\tau^{(k)}>t_i\}}\right)$. Ceci définit une procédure *Nested Monte Carlo* (NMC), ou *simulations dans les simulations*, pour simuler l'éspérance conditionnelle dans (1). Il est à noter qu'on aurait besoin d'une couche supplémentaire de simulations imbriquées si le MtM n'est pas analytique. Bien que ce schéma brute-force puisse être implémenté pour la CVA, il devient hors de portée pour la FVA.

En reprenant la notation précédente, et, uniquement dans un souci de simplicité pour cette introduction, en négligeant la plupart des termes de *rétroaction* et la fongibilité du capital à risque avec la marge de variation [Crépey et al., 2020], on peut écrire pour la FVA la définition simplifiée suivante:

$$\mathrm{FVA}_i := \mathbb{E}\left[\sum_{j=i}^{n-1}\bar{\gamma}_{j+1}\left(\mathrm{MtM}_{j+1}\mathbb{1}_{\{\tau>t_{j+1}\}}-\mathrm{CVA}_{j+1}-\mathrm{FVA}_{j+1}\right)^+(t_{j+1}-t_j)\middle| X_i,\mathbb{1}_{\{\tau>t_i\}}\right] \quad (2)$$

où $\bar{\gamma}$ est un processus stochastique représentant le spread de financement de la banque et est une composante du vecteur X. Ici, la FVA représente le coût de financement de notre transaction non collatéralisée et notre définition tient compte du fait que la CVA et la FVA elles-mêmes aident à réduire le besoin en financement. Modéliser la FVA comme nous l'avons fait permet également de tenir compte de ses fluctuations futures. Très récemment, trois banques américaines ont subi une perte combinée de 2 milliards de dollars en FVA [Becker, 2020] en raison de l'impact de la pandémie de COVID-19 et des réponses des banques centrales à celle-ci sur les taux d'intérêt et les spreads de financement (voir Tableau 1).

Ténor	**USD**	**EUR**
2 ans	226%	398%
5 ans	119%	170%
10 ans	83%	69%

Table 1. Sauts dans les spreads de financement du 2020-02-21 au 2020-03-24. Source: Takei, 2020 (IHS Markit).

Comme on peut le voir à partir de la définition dans (2), implémenter une procédure nested Monte Carlo pure pour la FVA comme nous l'avons présenté pour la CVA supposerait d'implémenter des simulations imbriquées avec autant de couches de sous-simulation qu'il y a de pas de temps jusqu'à maturité à cause de la dépendance de l'intégrand des valeurs futures de la FVA. Ceci est bien sûr prohibitif en terme de temps de calcul et conduirait à des algorithmes avec une complexité exponentielle en nombre de pas de temps. Cette complexité est encore exacerbée lorsqu'on prend en compte d'autres XVAs ou mesures de risque comme la KVA ou le capital économique compte tenu du couplage qu'ils créent notamment avec la FVA (voir Chapitre 1).

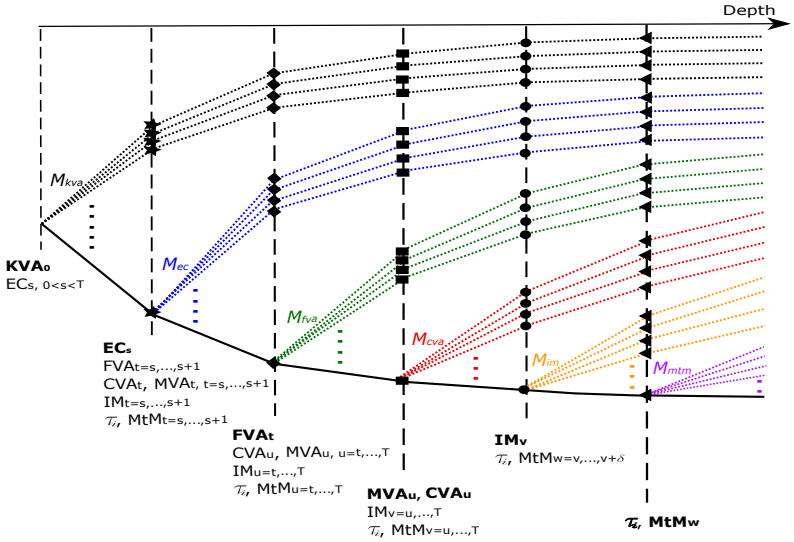

Figure 3. Interdépendance entre les différentes XVAs et mesures de risque dans un contexte nested Monte Carlo. Dans un tel contexte, pour simuler une réalisation du capital économique par exemple à un nœud Monte Carlo donné, il faut avoir des simulations internes de la FVA, CVA et MVA conditionnelles à ce nœud. Chacune de ces *FVAs intérieures* à son tour nécessite une autre couche de simulations internes et ainsi de suite. Source: Abbas-Turki et al., 2018.

Dans [Abbas-Turki et al., 2018], les auteurs proposent une approche benchmark impliquant plusieurs couches de nested Monte Carlo et des régressions linéaires, ainsi que des stratégies d'optimisation spécifiques aux GPUs. Ce benchmark a cependant une complexité exponentielle en nombre de couches XVA. Dans les chapitres 1, 2 et 4, nous développons une approche utilisant le Machine Learning, et plus précisément des régressions basées sur les réseaux de neurones, afin de rendre la complexité linéaire en nombre de couches XVA.

1.3 Des projections et non pas des ajustements de courbes

Comme introduites dans (1) et (2), chacune de la CVA et de la FVA est une espérance conditionnelle d'un certain intégrand. Le principe de base de l'approche que nous proposons est d'interpréter ces espérances conditionnelles comme des projections orthogonales de leurs intégrands respectifs. Plus généralement, supposons qu'on dispose de deux variables aléatoires X et Y telles que X soit supportée sur un espace $\mathcal{X}$ et Y soit de carré intégrable et supportée sur $\mathbb{R}$ et supposons qu'on s'intéresse à $\mathbb{E}[Y|X]$. Puisque $\mathbb{E}[Y|X]$ peut être vue comme une projection orthogonale de Y sur le sous-espace vectoriel composé de[2] variables aléatoires $\sigma(X)$-mesurables de carré intégrable, et donc un minimiseur de l'erreur de projection associée, on peut écrire:

$$\mathbb{E}[Y|X] = h^\star(X)$$

avec $h^\star$ est tel que

$$h^\star \in \underset{h\in\mathcal{B}(\mathcal{X},\mathbb{R})}{\operatorname{argmin}}\, \mathbb{E}[(h(X)-Y)^2] \tag{3}$$

2. plus précisément, des classes d'équivalence par rapport à la relation d'égalité presque sûre.

où $\mathcal{B}(\mathcal{X},\mathbb{R})$ est l'espace des fonctions boréliennes $h:\mathcal{X}\to\mathbb{R}$ telles que $h(X)$ est de carré intégrable. On peut aussi aboutir au même résultat de manière purement probabiliste comme suit:

$$\mathbb{E}[(h(X)-Y)^2]=\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2]+\underbrace{\mathbb{E}[\mathrm{var}(Y|X)]}_{\text{indépendant de } h},$$

où h est une fonction quelconque dans $\mathcal{B}(\mathcal{X},\mathbb{R})$.

La stratégie commune alors dans cette thèse est de résoudre le problème de minimisation de (3) en introduisant deux couches d'approximation:

- **Approximation de** $\mathrm{argmin}_{h\in\mathcal{B}(\mathcal{X},\mathbb{R})}$: nous proposons d'abord d'effectuer la minimisation dans un espace *suffisamment riche* de fonctions paramétrées pour que la minimisation devienne de dimension finie et puisse ainsi se faire par rapport aux paramètres via des techniques d'optimisation numérique classiques. Notre choix pour l'ensemble de la thèse s'est porté sur les réseaux de neurones (cf. chapitres 2, 3, 4 ou 5 pour dls définitions précises des réseaux de neurones adoptés dans cette thèse) et est motivé par les considérations suivantes:
  - **Propriété d'approximation universelle:** Si α est une fonction d'activation continûment différentiable non affine, $q\in\mathbb{N}^\star$ et $K\subset\mathbb{R}^n$ un compact. Alors pour tout $\varepsilon>0$ et $f\in\mathcal{C}(K,\mathbb{R}^q)$, il existe un réseau de neurones $u:\mathbb{R}^n\to\mathbb{R}^q$ avec p couches cachées, $q+n+2$ neurones par couche cachée et α comme fonction d'activation tel que $\sup_{x\in K}\|u(x)-f(x)\|<\varepsilon$. Ce résultat [Kidger and Lyons, 2020] est une version *profonde* (*i.e.* à largeur fixe) du théorème usuel d'approximation universelle à profondeur fixe pour les réseaux de neurones [Hornik, 1991; Cybenko, 1989]. Ainsi, on peut être optimiste quant à la possibilité de pouvoir se rapprocher de $h^\star$ en choisissant un réseau de neurones suffisamment grand soit en largeur soit en profondeur, avec des résultats récents comme [Cohen et al., 2016; Eldan and Shamir, 2016] plus en faveur des réseaux profonds;
  - **Capacité à *apprendre* automatiquement une base de régression linéaire:** En supposant qu'il n'y a pas de non-linéarité à la sortie du réseau de neurones illustré dans la Figure 4, la dernière couche cachée, lorsqu'elle est vue comme une fonction Φ_θ de l'entrée x paramétrée par la collection θ de tous les poids cachés, peut être interprétée comme une base de régression linéaire parametrisée par θ. Pour θ fixé, en désignant par w les poids de la couche de sortie et en supposant que la sortie est scalaire, alors entraîner un réseau de neurones en optimisant uniquement par rapport à w revient à effectuer une simple régression linéaire en utilisant $\Phi_\theta(x)$ comme base de régression. Ainsi, l'entraînement par rapport à tous les poids (*i.e.* θ et w conjointement) revient à apprendre à la fois la base de régression et les poids associés dans la couche de sortie. Ainsi, avec suffisamment de données, un réseau de neurones peut apprendre automatiquement une base de régression linéaire et ainsi éviter la sélection manuelle de celle-ci qui est généralement effectuée par des experts du domaine et qui ne peut pas être effectuée par exemple dans le cas des XVAs où la dépendance des XVAs des nombreux facteurs de risque est non triviale. Ceci rejoint l'idée de Samuel, 1959 ci-dessus qu'un programme peut être fait pour apprendre à résoudre une tâche, dans ce cas la sélection d'une base de régression, éventuellement plus efficacement que son programmeur;

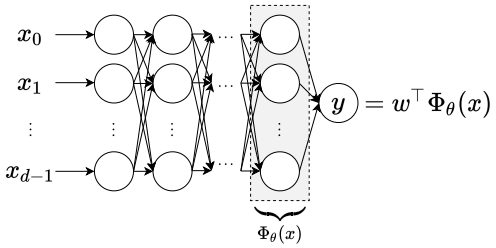

Figure 4. Entraîner en optimisant par rapport aux poids dans un réseau de neurones revient à apprendre à la fois la base de régression et les poids associés dans la couche de sortie.

- **Facilité de transfert de connaissances:** Nous pouvons faire un soi-disant *apprentissage par transfert* (*transfer learning* en anglais) [Bozinovski, 2020; Pan and Yang, 2009] avec des réseaux de neurones en réutilisant simplement les poids des entraînements de réseaux de neurones antérieurs et connexes. Nous avons beaucoup utilisé cette technique dans nos applications XVA dans les chapitres 2 et 4, où les tâches d'apprentissage associées à des pas de temps voisins sont nécessairement proches.

- **Approximation de** $\operatorname{argmin} \mathbb{E}[.]$: La minimisation de l'espérance de la perte ponctuelle (dans cet exemple, le carré de la distance entre la sortie du réseau de neurones et la variable aléatoire à projetter) est faite avec les outils de l'optimisation stochastique [Shapiro et al., 2021] en utilisant une approximation à échantillon fini de l'espérance. Cette approche est aussi connue sous le nom de *minimisation de risque empirique* (*empirical risk minimization* en anglais) dans le contexte de l'apprentissage statistique [Vapnik, 1991]. En particulier, nous résolvons numériquement le problème de minimisation en utilisant l'algorithme de la descente du gradient stochastique (plus précisément, en utilisant ADAM [Kingma and Ba, 2014]) sur l'approximation empirique[3]. Ici, nous tirons parti du fait que nous avons accès au processus de génération de données puisque les facteurs de risque sont déterminés par des équations différentielles stochastiques (EDS) connues. Les gradients par rapport aux paramètres du réseau de neurones sont calculés de manière exacte par *différentiation automatique* [Baydin et al., 2018; Savine, 2018], qui est implémentée par des librairies telles que JAX [Bradbury et al., 2018], PyTorch [Paszke et al., 2019] et Tensorflow [Abadi et al., 2016]. Nous référons aux chapitres 2 et 4 pour les pseudo-codes détaillés des procédures d'entraînement que nous avons utilisées.

Supposons maintenant qu'on a terminé l'approximation et qu'on dispose d'un candidat $h \in \mathcal{B}(\mathcal{X}, \mathbb{R})$. Une façon de mesurer si ce candidat est satisfaisant est de voir s'il est suffisamment proche (en fonction bien sûr d'un certain seuil subjectif de l'utilisateur de l'approche) de la *vraie valeur* (*i.e.* $\mathbb{E}[Y|X]$) en calculant la distance $\sqrt{\mathbb{E}[(h(X) - \mathbb{E}[Y|X])^2]}$. Cependant, l'utilisateur potentiel de nos approches d'apprentissage n'a généralement accès qu'à des couples de réalisations (X, Y) et n'a pas d'accès direct à $\mathbb{E}[Y|X]$, sinon il n'y

3. Cela signifie que nous ne tirons pas de réalisations complètement nouvelles à chaque itération SGD. Nous effectuons à la place plusieurs *passes* ou *epochs* sur l'ensemble de données, où à l'intérieur de chaque époque nous itérons sur des *mini-batchs* disjoints. Pour chaque mini-batch nous calculons le gradient de perte empirique en faisant la moyenne sur le mini-batch et nous effectuons une descente dans la direction opposée de ce gradient. L'objectif est de ne pas avoir à passer trop de temps sur de nouvelles simulations à chaque étape de SGD.

aurait pas besoin de l'approximer. On peut donc certes estimer $\mathbb{E}[(h(X)-Y)^2]$ directement mais non pas, du moins pas via Monte Carlo direct, $\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2]$. Bien sûr ces deux distances au carré, tout en ayant le même argmin par rapport à h sur $\mathcal{B}(\mathcal{X},\mathbb{R})$, ne sont pas nécessairement interchangeables (voir Figure 5).

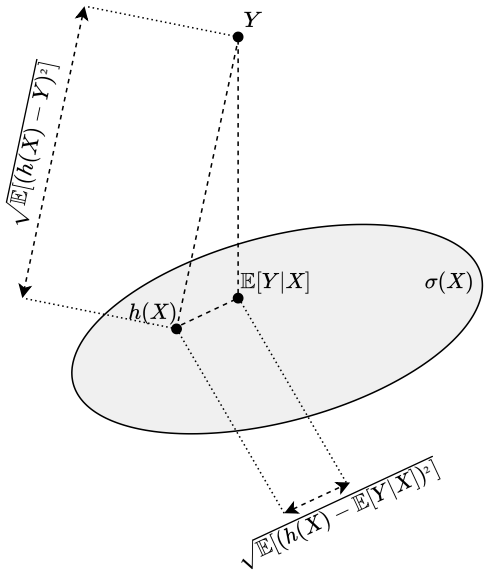

Figure 5. Les distances $\sqrt{\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2]}$ et $\sqrt{\mathbb{E}[(h(X)-Y)^2]}$ ne sont pas interchangeables en général, sauf si Y est $\sigma(X)$-mesurable.

Une façon d'estimer cette distance serait d'approximer $\mathbb{E}[Y|X]$ en utilisant une approche de Nested Monte Carlo. Cependant, cela est lent et inefficace, et ne convient pas à la validation en direct dans un environnement de production. À la place, nous avons développé un moyen simple dans le Chapitre 2 pour estimer notre distance L^2. En particulier, nous montrons que:

$$\mathbb{E}[(h(X)-\mathbb{E}[Y|X])^2] = \mathbb{E}[h(X)\,(h(X)-Y^{(1)}-Y^{(2)})+Y^{(1)}Y^{(2)}] \tag{4}$$

où $Y^{(1)}$ et $Y^{(2)}$ sont deux copies de Y indépendantes conditionnellement à X. D'où $h(X)\,(h(X)-Y^{(1)}-Y^{(2)})+Y^{(1)}Y^{(2)}$ est un estimateur sans biais de notre distance L^2 au carré et il ne nécessite que deux sous-simulations (par opposition à des centaines ou même des milliers si l'on devait approximer $\mathbb{E}[Y|X]$ directement en utilisant Nested Monte Carlo).

Nous étendons l'étude de cette approche d'apprentissage à l'approximation des Value-at-Risk et Expected-Shortfall conditionnels au Chapitre 3.

1.4 Non-stationnarité, accès au processus générateur des données

Nous soulignons une différence clé par rapport à la façon dont l'apprentissage automatique est utilisé dans d'autres disciplines. Alors que dans ces disciplines, on dispose généralement d'un ensemble de données fini, ces données provenant d'une collecte et d'une annotation manuelles ou automatiques, dans le cas de l'apprentissage à partir de données simulées, nous avons par définition un accès complet au processus de génération de données. Dans

notre cas, les facteurs de risque sont généralement modélisés à l'aide d'EDS ou de chaînes de Markov discrètes. Par conséquent, notre ensemble de données est généralement construit via des simulations de Monte Carlo des trajectoires de ces processus stochastiques et en calculant le long de chaque chemin l'intégrand que nous souhaitons projeter en utilisant des étapes algorithmiques précises. Cela rend les problèmes tels que l'*over-fitting* moins critiques, car nous pouvons toujours générer plus de données si nécessaire. Ces problèmes persistent en théorie et il convient toujours de vérifier les résultats à l'aide de métriques hors échantillon, calculées à l'aide d'un échantillon qui est indépendant des données d'apprentissage et qui est généralement appelé un *ensemble de test*.

Les espérances conditionnelles ou mesures de risque à approximer sont à calculer au moins quotidiennement. Cependant, les paramètres de modèle (par exemple les paramètres des EDS régissant les facteurs de risque) et d'autres variables, telles que la composition du portefeuille de la banque dans le cas de nos XVAs à l'échelle du portefeuille global, changent continuellement, introduisant ainsi une certaine non-stationnarité. Par conséquent, on ne peut pas simplement entraîner les réseaux de neurones approchants une fois puis les réutiliser en mode inférence dans toutes les conditions de marché et à n'importe quelle date ultérieure. À la place, il faut régénérer l'ensemble de données en lançant de nouvelles simulations Monte Carlo sous les nouveaux paramètres de modèle, puis réentraîner les réseaux de neurones en utilisant le nouvel ensemble de données et, dans le cas des XVAs, le nouveau portefeuille de la banque.

Dans ce contexte, la génération du jeu de données (*i.e.* les trajectoires Monte Carlo) et l'entraînement font toutes deux partie du *produit final*. C'est précisément pourquoi nous déployons des efforts considérables pour concevoir des simulations rapides. Nous utilisons en particulier des noyaux CUDA hautement optimisés pour les simulations de Monte Carlo et pour aider à la génération des *annotations* ou *labels* pour l'entraînement. Pour les BSDEs, nous nous servons des schémas d'apprentissage qui exploitent de nombreuses caractéristiques telles que la réutilisation des poids à travers les pas de temps (chapitres 2 et 4). Nous nous servons également de manière intensive du mécanisme de compilation juste-à-temps (JIT) de PyTorch. Dans le Chapitre 5, où nous n'avons pas ce problème de non-stationnarité mais où l'entraînement rapide est toujours souhaitable afin d'itérer plus efficacement dans la recherche et le développement, nous fournissons une application où nous sommes allés au-delà de ces optimisations et avons implémenté un entraînement haute performance directement en `C++` en utilisant la librairie `libtorch`.

1.5 Questions génériques connexes

Certaines questions ont été volontairement omises au cours de cette thèse car nous pensons que ces questions sont déjà abordées ou sont en train d'être traitées par d'autres chercheurs.

En particulier, alors que nous implémentons nos schémas d'apprentissage en utilisant un seul GPU, une implémentation réussie à grande échelle devra nécessairement utiliser plusieurs GPUs et même plusieurs nœuds.

Côté simulation, les différents GPUs ou nœuds n'ont pas besoin de communiquer entre eux et ils peuvent parfaitement générer leurs propres trajectoires Monte Carlo indépendamment les uns des autres en leur fournissant le code de simulation mais en configurant les générateurs de nombres aléatoires parallèles de telle manière à ce que l'indépendance des nombres aléatoires générés soit garantie entre les GPU et les nœuds [Abbas-Turki et al., 2014].

Côté entraînement, le bloc fonctionnel le moins trivial est l'algorithme SGD. Une façon simple et naïve de le paralléliser sur plusieurs unités de calcul, ou *worker*, est de commencer par les synchroniser[4], puis diviser et répartir sur ces workers le mini-batch[5] sur lequel la moyenne doit être calculée à une itération SGD donnée et demander à chaque worker de calculer les valeurs sur le batch puis leur moyenne, calculer les gradients associés puis renvoyer les résultats à un seul worker *maître* pour agrégation finale et laisser tous les autres workers au repos pendant que le worker maître met à jour les paramètres du réseau de neurones à l'aide d'une étape SGD. Ces synchronisations et ces verrous créent une surcharge qui rend la parallélisation entre plusieurs unités de calcul moins attrayante. Plus récemment, des approches qui évitent le verrouillage et la synchronisation tout en ayant des garanties théoriques ont vu le jour et nous citons notamment l'algorithme Hogwild! [Recht et al., 2011] qui est nativement implémenté par `PyTorch`. Nous référons à [Chen et al., 2016] pour une discussion détaillée des avantages et des inconvénients des SGD synchrones et asynchrones dans le contexte de l'apprentissage distribué.

Un autre problème est celui de l'optimisation des hyper-paramètres, par exemple le nombre de couches et de neurones dans un réseau. Bien que nous ne pensons pas que ce soit quelque chose qui doive être faite systématiquement à chaque exécution, à moins que les paramètres de modèle et les conditions du marché ne présentent de forts changements, nous pensons qu'il s'agit d'un *fine-tuning* qui pourrait être effectué occasionnellement dans le cadre du maintien des codes d'approximation du réseau de neurones. Les approches exhaustives et naïves incluent le *Grid Search*, où on teste toutes les combinaisons dans une grille discrète d'hyper-paramètres et on choisit celles qui minimisent l'objectif d'entraînement (ou une autre fonction de coût pertinente, selon la tâche). Ce dernier est évalué à l'aide d'un jeu de données, généralement appelé *ensemble de validation*, qui est indépendant de celui qui a été utilisé pour l'entraînement mais également indépendant de l'ensemble de test. Cette approche, bien que très simple à mettre en œuvre, est lente et souffre de la malédiction de la dimensionnalité en raison de la nécessité de construire une grille de valeurs de paramètres. Des approches plus intelligentes sont basées sur l'optimisation bayésienne [Shahriari et al., 2015] et incluent des algorithmes basés sur les bandits [Slivkins, 2019] tels que HyperBand [Li et al., 2017], pour lesquels nous référons à `Optuna` [Akiba et al., 2019] pour une implémentation professionnelle et stable compatible avec `PyTorch`.

Enfin, nous reconnaissons que l'idée de devoir investir dans des infrastructures, en particulier des clusters GPU, afin de mettre en œuvre nos approches proposées dans des contextes à grande échelle peut sembler intimidante. Cependant, grâce à la disponibilité de solutions cloud telles qu'Amazon AWS ou Google Cloud pour n'en nommer que quelques-unes, on peut *commencer petit* avec des prototypes pour évaluer à la fois l'applicabilité et le retour sur investissement auquel on peut s'attendre, puis évoluer au besoin avant d'investir dans de grandes infrastructures internes.

2 Résumés des chapitres

Les notations mathématiques dans cette section sont locales à chaque sous-section uniquement.

4. *i.e.* s'assurer qu'ils ont tous les mêmes copies des paramètres du réseau de neurones.

5. Aucune communication de données n'est nécessaire dans notre cas car chaque worker est déjà censé avoir sa propre instance du code de simulation générant des trajectoires Monte Carlo parfaitement indépendantes.

2.1 Chapitre 1 – XVA Analysis From the Balance Sheet

La crise financière de 2008–2009 a changé la manière dont les produits dérivés sont valorisés. Cela a poussé les banques à prendre de plus en plus en compte les ajustements de valeur, appelés XVAs comme introduit ci-dessus. Ceux-ci rendent la valorisation davantage non linéaire et nécessitent une approche globale de celle-ci, c'est-à-dire la prise en compte de l'ensemble de leur portefeuille. Le modèle de structure du capital et l'approche d'ajustement de valeur décrits dans ce chapitre sont ancrés dans une perspective de bilan et de politique de dividende. Cette approche est alignée sur l'intérêt des actionnaires et donne un sens économique précis aux différents termes XVA.

Dans le modèle de structure de capital proposé, nous distinguons principalement les *desks contra-asset (CA)*, qui sont en charge du risque de contrepartie et de ses implications de financement, et les *clean desks*, qui se concentrent sur les risques de marché liés à leurs lignes de métier respectives. Les desks CA évaluent les contra-assets (*i.e.* CVA et FVA), les imputent aux clients de manière incrémentielle et effectuent des dépôts sur un compte de réserve qui est ensuite utilisé pour faire face aux pertes moyennes dues au risque de contrepartie et aux dépenses de financement. Une autre partie importante dans le modèle de structure du capital est la direction qui génère une marge de risque. Celle-ci est ensuite versée sous la forme de paiements de KVA aux actionnaires, en tant que prime de risque sur leur *capital à risque*, *i.e.* la différence[6] entre un Expected-Shortfall de la perte du desk CA et la KVA, supposée être absorbante pour les pertes.

Sous la forme la plus simple, *i.e.* en supposant[7] des couvertures entièrement collatéralisées, aucun recouvrement en cas de défaut et aucune marge de variation sur les transactions avec les clients, une base stochastique de valorisation risque-neutre[8] $(\Omega, \mathcal{A}, \mathcal{F}, \mathbb{Q})$ avec opérateur d'espérance $\mathbb{E}$ et $\mathbb{E}_t = \mathbb{E}[.|\mathcal{F}_t]$, le modèle de structure du capital proposé donne lieu aux équations XVA suivantes en temps continu[9]:

$$
\begin{aligned}
\mathrm{CVA}_t &= \sum_{c\in\mathcal{C}} \mathbb{E}_t\Big[\int_t^T \big(\mathrm{MtM}_s^{(c)}\big)^+ \delta_{\tau^{(c)}}(\mathrm{d}s)\Big] \\
\mathrm{FVA}_t &= \mathbb{E}_t\Big[\int_t^T \bar{\gamma}_s \Big(\sum_{c\in\mathcal{C}} \mathrm{MtM}_s^{(c)} \mathbb{1}_{\{\tau^{(c)}>s\}} - \mathrm{CA}_s - \mathrm{CR}_s\Big)^+ \mathrm{d}s\Big] \\
\mathrm{CA}_t &= \mathrm{CVA}_t + \mathrm{FVA}_t \\
\mathrm{CR}_t &= \max\left(\mathrm{EC}_t, \mathrm{KVA}_t\right) \\
\mathrm{EC}_t &= \mathbb{ES}_t[L_{t+1} - L_t] \\
\mathrm{d}L_t &= \mathrm{dCA}_t + \underbrace{\sum_{c\in\mathcal{C}} \big(\mathrm{MtM}_t^{(c)}\big)^+ \delta_{\tau^{(c)}}(\mathrm{d}t)}_{\text{default losses}} + \underbrace{\gamma_t \Big(\sum_{c\in\mathcal{C}} \big(\mathrm{MtM}_t^{(c)}\big)^+ \mathbb{1}_{\{\tau^{(c)}>t\}} - \mathrm{CA}_t - \mathrm{CR}_t\Big)^+ \mathrm{d}t}_{\text{funding expenditures}} \\
\mathrm{KVA}_t &= \mathbb{E}_t\Big[\int_t^T h\left(\mathrm{CR}_s - \mathrm{KVA}_s\right)^+ \mathrm{d}s\Big]
\end{aligned}
$$

6. Plus précisément la partie positive de la différence, *i.e.* $(\mathrm{EC} - \mathrm{KVA})^+$ où EC est le Capital Èconomique (*Economic Capital* en anglais), défini comme un Expected-Shortfall à 97.5% de la perte du desk CA.

7. Tous ces éléments sont cependant pleinement pris en compte dans le chapitre complet.

8. $\mathbb{Q}$ ici est la mesure de survie de la banque.

9. Ces équations à temps continu sont en fait un cas particulier d'équations différentielles stochastiques rétrogrades anticipées (ABSDE) pour lesquelles nous donnons un schéma général d'apprentissage au Chapitre 4.

où T est la maturité du portefeuille de la banque (supposé détenu sur une base *run-off*), CVA_t, FVA_t, CA_t, CR_t, EC_t, KVA_t et L_t sont les valeurs à l'instant t des processus respectifs CVA, FVA, contra-assets (*i.e.* CVA + FVA), capital-at-risk, capital économique, KVA et perte du desk CA. $\mathcal{C}$ est un ensemble fini indexant les contreparties de la banque, et pour chaque $c \in \mathcal{C}$, $\mathrm{MtM}^{(c)}$ et $\tau^{(c)}$ sont respectivement le processus mark-to-market de la banque des positions avec la contrepartie c et le temps de défaut de celle-ci, supposé être un temps d'arrêt par rapport à $\mathcal{F}$. Le processus γ est le spread de financement de la banque et h un taux de rendement minimal constant représentant le taux auquel les actionnaires s'attendent à être payés pour leur capital à risque.

$\mathbb{ES}_t$ est l'opérateur Expected-Shortfall conditionnel, que nous définissons comme suit pour toute variable aléatoire ℓ $\mathcal{F}_T$-mesurable:

$$\mathbb{ES}_t[\ell] = \mathbb{E}_t[\ell \,|\, \ell \geq \mathbb{V}a\mathbb{R}_t(\ell)]$$

et $\mathbb{V}a\mathbb{R}_t$ est le quantile gauche conditionnel à $\mathcal{F}_t$ de ℓ au niveau α, que nous allons aussi appeler par la suite Value-at-Risk (VaR) conditionnelle au niveau de confiance α.

Dans un cadre markovien, *i.e.* lorsque $\mathbb{E}_t[.] = \mathbb{E}[.|\mathcal{X}_t]$ et $\mathbb{V}a\mathbb{R}_t$ coïncide avec la VaR conditionnelle à $\mathcal{X}_t$ dans les équations ci-dessus pour un processus de facteurs de risque $\mathcal{X}$, nous proposons d'approximer les processus définis ci-dessus à l'aide de réseaux de neurones. Pour les processus définis à l'aide d'espérances conditionnelles, tels que la CVA, la FVA et la KVA ci-dessus, nous utilisons la régression par moindres carrés par rapport au processus de facteur de risque $\mathcal{X}$ avec des réseaux de neurones comme approximateurs. Pour l'Expected-Shortfall conditionnel, cependant, nous n'avons pas de résultat d'*élicitabilité* direct, *i.e.* la représentation fonctionnelle de l'Expected-Shortfall conditionnel n'est pas un minimiseur d'une certaine fonction de perte[10]. Cependant, d'après [Fissler and Ziegel, 2016; Fissler et al., 2016], le couple composé de l'Expected-Shortfall conditionnel et du quantile conditionnel est *conjointement élicitable*, *i.e.* on peut récupérer les deux en même temps en minimisant une certaine fonction de perte. Nous implémentons donc un algorithme d'apprentissage conjoint de l'Expected-Shortfall et Value-at-Risk conditionnels qui minimise cette fonction de perte sur un espace de réseaux de neurones avec des sorties dans $\mathbb{R}^2$ (*i.e.* c'est-à-dire produisant un couple de $\mathbb{ES}$ et $\mathbb{V}a\mathbb{R}$) par opposition aux réseaux de neurones à sorties scalaires usuels utilisés pour approcher des espérances conditionelles.

Enfin, nous effectuons des expériences numériques approfondies montrant que l'approche holistique proposée peut être implémentée numériquement. Les XVAs incrémentiels sont calculés efficacement pour chaque trajectoire Monte Carlo en utilisant une combinaison de réseaux de neurones, d'itérations de Picard et de calcul sur GPU.

2.2 Chapitre 2 – Pathwise CVA Regressions With Oversimulated Defaults

Dans ce chapitre, nous abordons les problèmes de variance potentiels découlant de la nécessité de simuler spécifiquement les défauts dans notre approche XVA au Chapitre 1. Dans le cas de la CVA, les praticiens contournent généralement ce problème en remarquant qu'on peut l'écrire sous *forme intensité* comme suit:

$$\begin{aligned}
\mathrm{CVA}_t &= \sum_{c\in\mathcal{C}} \mathbb{E}_t\Big[\int_t^T \big(\mathrm{MtM}_s^{(c)}\big)^+ \delta_{\tau^{(c)}}(\mathrm{d}s)\Big] \\
&= \sum_{c\in\mathcal{C}} \mathbb{E}_t\Big[\int_t^T \big(\mathrm{MtM}_s^{(c)}\big)^+ \gamma_s^{(c)} \exp\Big(-\int_t^s \gamma_u^{(c)}\,\mathrm{d}u\Big)\,\mathrm{d}s\Big]
\end{aligned}$$

10. à moins d'avoir accès au quantile conditionnel.

en supposant que chaque contrepartie $c \in \mathcal{C}$ a un processus d'intensité de défaut stochastique $\gamma^{(c)}$ tel que, pour tout $0 \leq t < s$,

$$\mathbb{Q}(\tau^{(c)} > s | \mathcal{F}_t, \{\tau^{(c)} > t\}) = \mathbb{E}_t\left[\exp\left(-\int_t^s \gamma_u^{(c)} \, \mathrm{d}u\right)\right]$$

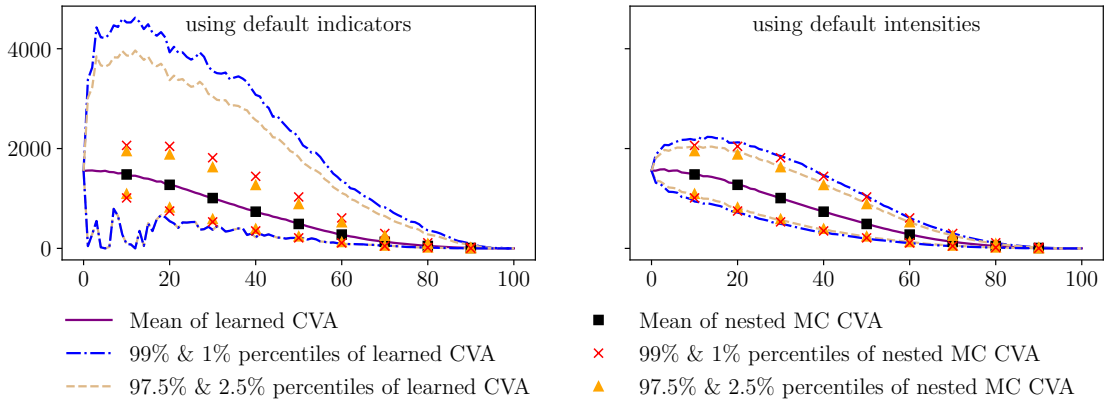

Figure 6. Les indicatrices de défaut étant désormais absentes de l'intégrand définissant notre CVA, les régressions se comportent *mieux* car la variance de l'intégrand est réduite. **abscisses:** pas de temps de valorisation, **ordonnées:** niveau de la statistique considérée de la CVA au pas de temps donné.

Cependant, cette solution de contournement basée sur l'intensité s'applique uniquement aux espérances conditionnelles avec des intégrands linéaires dans les indicatrices de défaut. En particulier, cela ne s'applique pas à la FVA qui comporte des indicatrices de survie à l'intérieur d'une non-linéarité dans l'intégrand. Ceci est également vrai pour l'EC dont la définition dépend d'un Expected Shortfall de pertes, incluant les pertes dues au défaut et les dépenses liées au financement qui à leur tour dépendent des indicatrices de survie.

Pour résoudre ce problème, nous proposons un schéma de simulation simple où nous séparons les facteurs de risque en deux sous-groupes: un groupe qui ne contribue pas significativement à notre problème de variance (en particulier les facteurs diffusifs) représenté par un processus vectoriel Y, et un autre qui est composé des principaux contributeurs à la variance (dans notre cas les indicatrices de défaut) et représentés par un autre processus vectoriel X. L'idée principale est alors de simuler, à chaque pas de temps, plus de réalisations de X conditionnelles à chaque réalisation de Y. Ceci est motivé par une simulation peu coûteuse des défauts conditionnellement aux facteurs de risque diffusifs.

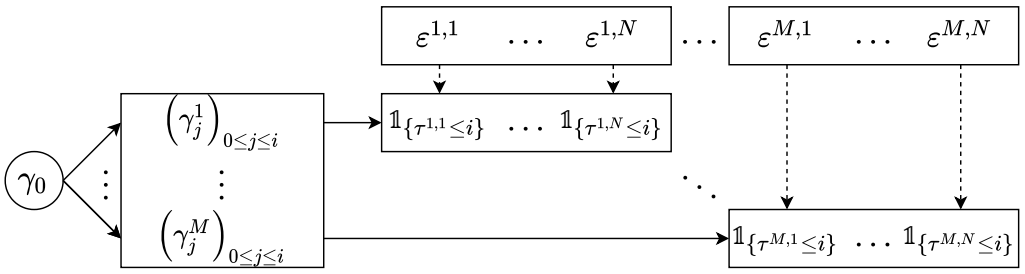

Figure 7. Schéma de simulation proposé pour les défauts.

En supposant pour simplifier dans ce résumé une seule contrepartie, et donc en omettant l'indice de la contrepartie dans la notation du processus d'intensité γ, et en supposant que tous les processus considérés sont maintenant en temps discret (*i.e.* γ_i est l'intensité de défaut en temps discret au i-ème pas de temps d'une certaine discrétisation temporelle), l'idée dans le cas de notre cadre XVA est de simuler d'abord un certain nombre M de trajectoires des facteurs de risque diffusifs (dont γ fait partie). Ensuite, à chaque pas de temps i, on suppose avoir accès aux trajectoires jusqu'au pas de temps i de γ, *i.e.* un échantillon i.i.d $\{(\gamma_j^k)_{0\leq j\leq i}\}_{1\leq k\leq M}$ de $(\gamma_j)_{0\leq j\leq i}$. On simule alors conditionnellement à chaque trajectoire, indexée par $k\in\{1,\dots,M\}$, N réalisations i.i.d de l'indicatrice de défaut, *i.e.* un échantillon $\{\mathbb{1}_{\{\tau^{k,l}\leq i\}}\}_{1\leq l\leq N}$ (en utilisant un échantillon i.i.d $\{\varepsilon^{k,l}\}_{1\leq l\leq N}$ d'une variable aléatoire exponentielle standard).

Cela permet alors de définir un échantillon de taille $M\times N$ de n'importe quel intégrand dans les équations XVA, ou toute autre tâche de régression qui rentre dans notre cadre, en considérant les différentes combinaisons de l'indice de conditionnement (k ci-dessus) et de l'indice de la *sur-simulation* (l ci-dessus). L'échantillon qui en résultera ne sera cependant pas composé de réalisations indépendantes. Mais nous montrons numériquement qu'un tel schéma d'échantillonnage est efficace pour résoudre le problème de variance décrit ci-dessus pour un faible coût de calcul.

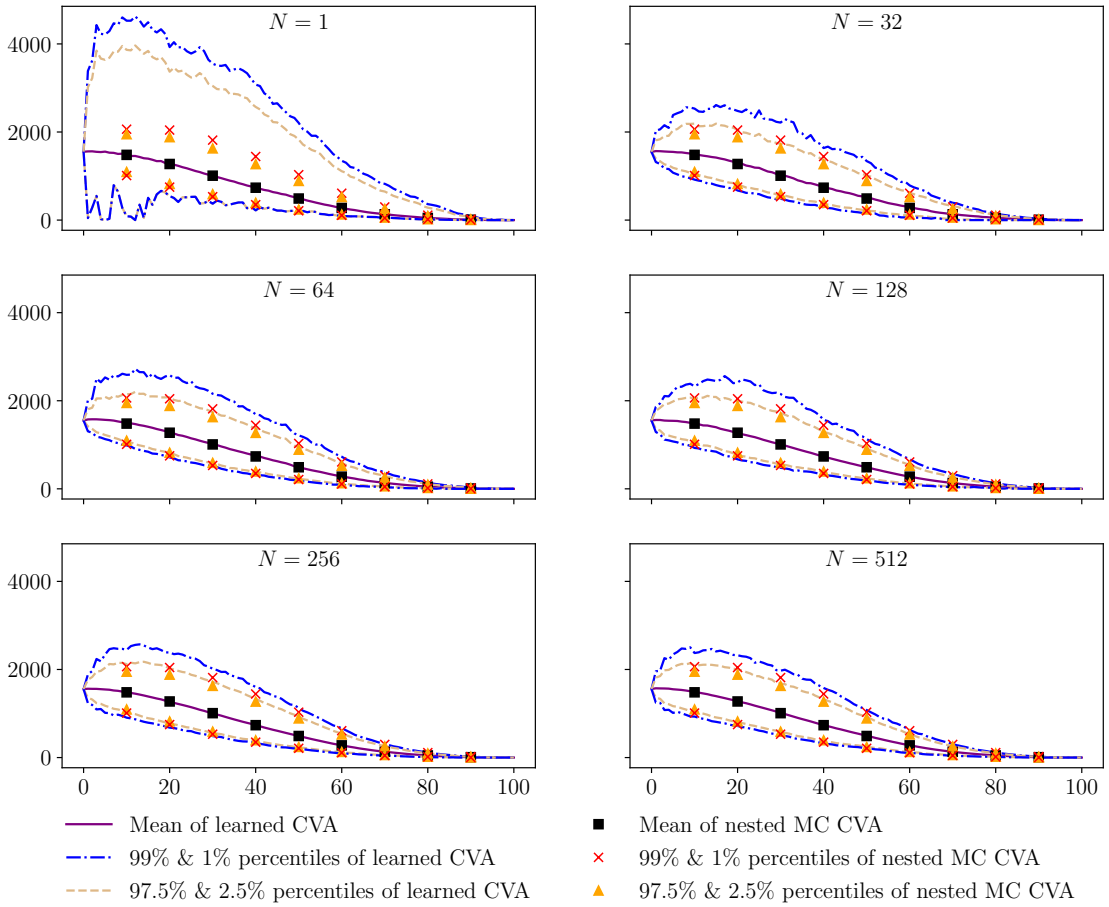

Figure 8. À partir de seulement $N=32$ simulations supplémentaires des défauts conditionnellement à chaque réalisation des facteurs de risque diffusifs (en supposant un échantillon de taille $M=2^{14}=16384$ des réalisations diffusives), nous obtenons déjà de meilleurs profils CVA par rapport à la situation sans sur-simulation des défauts (*i.e.* $N=1$). **abscisses:** pas de temps de valorisation, **ordonnées:** niveau de la statistique considérée de la CVA au pas de temps donné.

Nous étendons également les résultats du cadre de *l'approximation avec moyenne empirique* de [Shapiro et al., 2021] à notre cas non i.i.d et donnons des garanties de convergence statistique sous la forme d'une inégalité de déviation qui aide à comprendre comment M et N impactent la convergence du minimum de la perte empirique (où la moyenne est calculée sur nos $M \times N$ réalisations) vers minimum de la perte théorique, c'est-à-dire celle où nous utilisons l'opérateur d'espérance au lieu d'une moyenne sur un échantillon fini.

Enfin, dans le cas des espérances conditionnelles, nous abordons le problème de la validation de cette approche d'apprentissage dans un contexte de production en direct sans accès à un benchmark Nested Monte Carlo. Nous fournissons une procédure, introduite ci-dessus dans (4), pour estimer la distance L^2 entre l'approximation apprise et l'espérance conditionnelle recherchée, et qui ne nécessite aucune connaissance de cette dernière.

2.3 Chapitre 3 – Learning Value-at-Risk and Expected Shortfall

Dans ce chapitre, nous étudions une approche en deux étapes pour apprendre la Value-at-Risk et l'Expected Shortfall conditionnels. Considérons un espace probabilisé $(\Omega, \mathcal{A}, \mathbb{P})$ et soient X un vecteur aléatoire supporté sur un espace polonais S et Y une variable aléatoire réelle intégrable. Si nous définissons la $\mathbb{V}a\mathbb{R}$ et l'$\mathbb{ES}$ comme suit:

$$\begin{aligned} \mathbb{V}a\mathbb{R}(Y|X) &= \inf\{y \in \mathbb{R} : \mathbb{P}(Y \leq y|X) \geq \alpha\} \\ \mathbb{ES}(Y|X) &= \frac{1}{1-\alpha}\mathbb{E}[Y\,\mathbb{1}_{\{Y \geq \mathbb{V}a\mathbb{R}(Y|X)\}}|X] \end{aligned}$$

pour un certain niveau de confiance $\alpha \in (0,1)$, alors il existe des fonctions boréliennes q et s telles que

$$\begin{aligned} \mathbb{V}a\mathbb{R}(Y|X) &= q(X) \\ \mathbb{ES}(Y|X) &= s(X) \end{aligned}$$

En outre,

$$\begin{aligned} q &\in \operatorname*{argmin}_{f \in \mathcal{B}^1(S)} \mathbb{E}\left[\frac{1}{1-\alpha}(Y - f(X))^+ + f(X)\right] \\ s &\in \operatorname*{argmin}_{f \in \mathcal{B}^2(S)} \mathbb{E}[(Y\,\mathbb{1}_{\{Y \geq q(X)\}} - f(X))^2] \end{aligned}$$

où $\mathcal{B}^1(S)$ et $\mathcal{B}^2(S)$ sont les ensembles de fonctions boréliennes $f : S \to \mathbb{R}$ tels que $f(X)$ est respectivement intégrable et de carré intégrable. Nous donnons des fonctions de perte plus générales dans le Theorème 3.5 de ce chapitre. La Value-at-Risk et l'Expected Shortfall conditionnels peuvent ensuite être appris via l'*approche en deux étapes* suivante:

1. Tirer un échantillon i.i.d $\{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ de (X, Y);

2. Apprendre $\mathbb{V}a\mathbb{R}(Y|X)$ en cherchant $\hat{q}$ tel que

$$\hat{q} \in \operatorname*{argmin}_{f \in \mathcal{F}} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{1-\alpha}(Y_i - f(X_i))^+ + f(X_i);$$

3. Apprendre $\mathbb{ES}(Y|X)$ en cherchant $\hat{s}$ tel que

$$\hat{s} \in \underset{f \in \mathcal{G}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \left(\hat{q}(X_i) + \frac{1}{1-\alpha} (Y_i - \hat{q}(X_i))^+ - f(X_i) \right)^2.$$

où $\mathcal{F}$ et $\mathcal{G}$ sont deux familles de fonctions (aussi appelées *espaces d'hypothèses*) sur lesquelles on cherche une approximation respectivement de la Value-at-Risk et de l'Expected Shortfall conditionnels. La seconde étape approxime $\mathbb{ES}(Y|X) = \frac{1}{1-\alpha} \mathbb{E}[Y \mathbb{1}_{\{Y \geq \mathbb{V}a\mathbb{R}(Y|X)\}}|X]$ en remplaçant $\mathbb{V}a\mathbb{R}(Y|X)$ par le candidat appris $\hat{q}(X)$, *i.e.*

$$\begin{aligned} \mathbb{ES}(Y|X) &= \mathbb{V}a\mathbb{R}(Y|X) + \frac{1}{1-\alpha} \mathbb{E}[(Y - \mathbb{V}a\mathbb{R}(Y|X))^+|X] \\ &\approx \hat{q}(X) + \frac{1}{1-\alpha} \mathbb{E}[(Y - \hat{q}(X))^+|X] \end{aligned}$$

Basée sur des résultats de la théorie de Rademacher et Vapnik-Chervonenkis [Shalev-Shwartz and Ben-David, 2014] et sur des bornes non-asymptotiques prouvées dans [Barrera, 2022], une analyse de convergence non-asymptotique est fournie. Le Theorème 3.21 énonce notamment que:

$$\begin{aligned} c_{B_1} \mathbb{E}_n[(\hat{q}(X) - q(X))^2] \leq & \left(2(2-\alpha) \inf_{f \in \mathcal{F}} \mathbb{E}[|f(X) - q(X)|] \right) \wedge \left(C_{B_1} \inf_{f \in \mathcal{F}} \mathbb{E}[(f(X) - q(X))^2] \right) \\ & + \frac{4(2-\alpha)B_2}{\sqrt{n}} \sqrt{2 \log\left(\frac{2}{\delta} \right)} \\ & + \frac{8(2-\alpha)B_1}{\sqrt{n}} \left(1 + \mathbb{E}\left[\sqrt{2 \log\left(N_1\left(\mathcal{F}, X_{1:n}, \frac{B_1}{\sqrt{n}} \right) \right)} \right] \right) \end{aligned}$$

pour tout $\delta \in (0,1)$ avec une probabilité d'au moins $1-\delta$, où $X_{1:n} = \{X_1, \ldots, X_n\}$, $\mathbb{E}_n[.] = \mathbb{E}[.|X_{1:n}]$ et nous supposons que $\mathcal{F}$ est uniformément borné par $B_1 > 0$, avec $\mathbb{V}a\mathbb{R}(Y|X)$ également supposée bornée par la même constante. On suppose aussi que Y est bornée par une certaine constante $B_2 > 0$ telle que $B_1 \leq B_2$ et qu'il existe des constantes $C_{B_1} \geq c_{B_1} > 0$ telles que

$$c_{B_1} \leq F'_{Y|X}(y) \leq C_{B_1}$$

$\mathbb{P}$-p.s pour tout $y \in [-B_1, B_1]$, où $F_{Y|X}$ est la fonction de répartition de Y conditionnellement à X. N_1 est un certain nombre de recouvrements défini plus en détail dans la Définition 3.17. Un résultat similaire est également montré pour l'approximation de l'écart entre $\mathbb{ES}(Y|X)$ et $\mathbb{V}a\mathbb{R}(Y|X)$, *i.e* la différence $s - q$, dans le Théorème 3.25.

Nous fournissons également plusieurs schémas d'apprentissage, utilisant des réseaux de neurones, pour approximer $\mathbb{V}a\mathbb{R}(Y|X)$ et $\mathbb{ES}(Y|X)$ pour plusieurs niveaux de confiance α en même temps. Parmi ces schémas, nous en proposons un nouveau où nous pénalisons la partie négative de la dérivée du réseau de neurones approchant la $\mathbb{V}a\mathbb{R}$ par rapport au niveau de confiance α, randomisé et considéré ici comme une covariable à côté de X. Nous assurons aussi numériquement une quasi-inexistence du problème du *croisement de quantiles* [Takeuchi et al., 2006; He, 1997; Koenker and Park, 1996].

Afin d'évaluer numériquement nos schémas d'apprentissage proposés, nous avons également réalisé deux expériences: l'apprentissage de $\mathbb{V}a\mathbb{R}(Y|X)$ et $\mathbb{ES}(Y|X)$ dans un cadre gaussien avec des moments conditionnels d'ordre 1 et 2 de Y polynomiaux en X, et un

cas d'étude plus complexe où nous apprenons une *marge initiale dynamique* dans un cadre XVA similaire au Chapitre 2. Pour les besoins de ce dernier exemple, nous fournissons une procédure de Nested Monte Carlo qui effectue un *apprentissage* non paramétrique de $\mathbb{V}a\mathbb{R}(Y|X)$ en utilisant des descentes du gradient stochastiques *conditionnelles* accélérées via une initialisation avec une Value-at-Risk gaussienne conditionnelle.

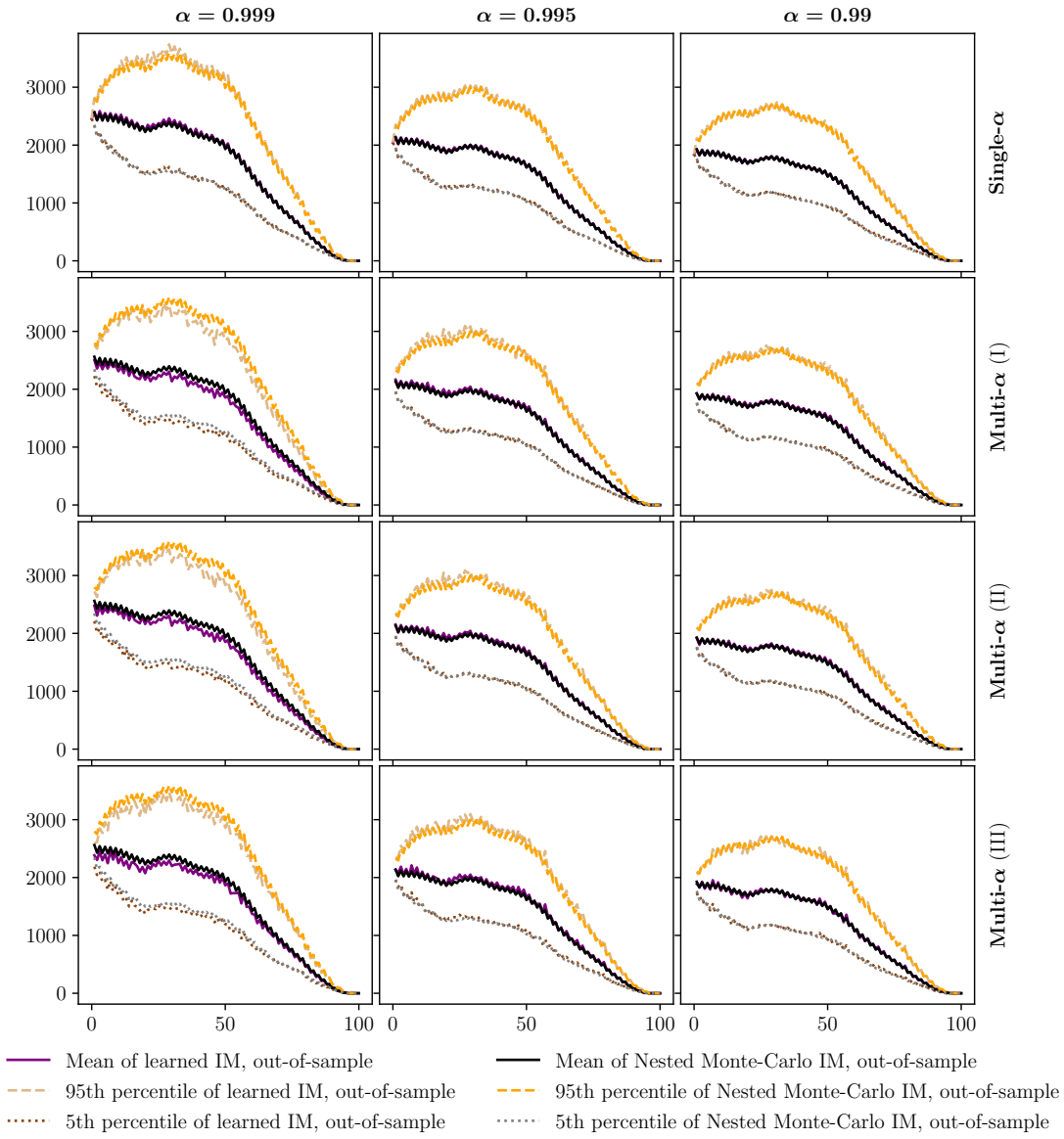

Figure 9. Nous obtenons avec succès les profils d'une marge initiale dynamique (IM) pour différents niveaux de confiance α dans notre cadre XVA. Chaque colonne représente un niveau de confiance donné α et chaque ligne représente un des schémas d'apprentissage détaillés au Chapitre 3. **Abscisses:** pas de temps de valorisation, **ordonnées:** niveau de la statistique considérée de l'IM au pas de temps donné.

Enfin, nous abordons également la question de la validation de cette approche d'apprentissage dans des contextes où on n'a pas accès aux valeurs de référence et où on ne peut pas se permettre le coût du calcul d'un benchmark Nested Monte Carlo. Nous étendons en particulier l'astuce du *twin simulation* introduite dans le Chapitre 2 à l'estimation de la

distance en p-values (resp. en L^2) entre l'approximation apprise et la valeur de référence $\mathbb{V}a\mathbb{R}(Y|X)$ (resp. $\mathbb{ES}(Y|X)$).

2.4 Chapitre 4 – Pathwise XVAs: The Direct Scheme

En supposant que le capital à risque est fongible pour la marge de variation, nous avons vu au Chapitre 1 que la FVA, à tout instant $0 \leq t \leq T$, peut s'écrire en temps continu comme suit:

$$\mathrm{FVA}_t = \mathbb{E}_t\left[\int_t^T \bar{\gamma}_s \left(\sum_{c\in\mathcal{C}} \mathrm{MtM}_s^{(c)} \mathbb{1}_{\{\tau^{(c)}>s\}} - \mathrm{CA}_s - \mathrm{CR}_s\right)^+ \mathrm{d}s\right]. \tag{5}$$

Cependant, nous avons $\mathrm{CA} = \mathrm{CVA} + \mathrm{FVA}$ et le terme de capital à risque CR dépend du capital économique EC qui est à son tour défini comme un Expected Shortfall des pertes futures du desk contra-assets. Ces dernières concernent les pertes dues à la variation de la FVA et aux dépenses de financement. L'équation (5) appartient à la classe des *équations différentielles stochastiques rétrogrades anticipées* (ABSDE) [Peng and Yang, 2009]. Crépey et al., 2020 montrent l'existence d'une solution unique à de telles ABSDEs lorsque le terme anticipatif dans le *driver* dépend d'un Expected Shortfall conditionnel d'incréments futurs de la partie martingale de la solution. Dans ce chapitre, nous nous intéressons à la résolution numérique des ABSDE de la forme:

$$Y_t = \mathbb{E}_t\left[\phi(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, \mathbb{ES}_s(\Phi_{\bar{s}}(M)))\,\mathrm{d}s\right] \tag{6}$$

où Y est une semi-martingale spéciale avec M comme composante martingale locale canonique dans sa décomposition Doob-Meyer, $\mathcal{X} = (X, J)$ avec X une solution forte à valeurs dans $\mathbb{R}^p$ d'une EDS et J une *pseudo-chaîne de Markov* à valeurs dans $\{0,1\}^q$, ϕ est une fonction continue de $\mathbb{R}^p$ dans $\mathbb{R}^l$, f est une fonction continue de $[0,T]\times\mathbb{R}^p\times\mathbb{R}^l\times\mathbb{R}$ dans $\mathbb{R}^l$ satisfaisant certaines hypothèses d'intégrabilité et de régularité Lipschitz, $\bar{\cdot}$ est un opérateur déterministe qui transforme chaque instant t en un instant $\bar{t} \in [t,T]$, $\mathbb{ES}_s$ est un Expected Shortfall conditionnel tel que défini précédemment dans le Chapitre 1, et $\Phi_{\bar{s}}$ est définie pour tout $s \in [0,T]$ comme suit:

$$\Phi_{\bar{s}}(M) = \Phi(s; \mathcal{X}_{[s,\bar{s}]}, M_{[s,\bar{s}]} - M_s)$$

où $\Phi(s; \mathbf{x}, \mathbf{m})$ est une application déterministe à valeurs réelles du temps s et des trajectoires càdlàg $\mathbf{x}$ et $\mathbf{m}$ sur $[s,\bar{s}]$ tels que $\mathbf{m}_s = 0$, et M est la partie martingale de Y.

Alors que nous avons résolu un cas particulier d'ABSDE dans le Chapitre 1 en utilisant une approche d'apprentissage combinée avec des itérations de Picard, nous proposons dans ce chapitre un schéma qui ne nécessite pas d'itérations de Picard, implémenté à l'aide d'approximations avec des réseaux de neurones, pour résoudre des ABSDEs plus générales de la forme de (6). Plus précisément, nous proposons le schéma explicite en temps discret suivant:

$$\begin{aligned} Y_{t_i}^h &= \mathbb{E}_{t_i}[Y_{t_{i+1}}^h + f(t_i, \mathcal{X}_{t_i}^h, Y_{t_{i+1}}^h, \rho_{t_{i+1}}^h)\,\Delta t_{i+1}] \\ \rho_{t_i}^h &= \mathbb{ES}_{t_i}\left(\Phi_{t_i}^h\left(Y_{t_\ell}^h + \sum_{k<\ell} f(t_k, \mathcal{X}_{t_k}^h, Y_{t_{k+1}}^h, \rho_{t_{k+1}}^h)\,\Delta t_{k+1}, \ell = 0,\ldots,n\right)\right) \end{aligned}$$

où nous avons considéré une grille de temps $0 = t_0 < t_1 < \cdots < t_n = T$, $\Delta t_{i+1} = t_{i+1} - t_i$, $\bar{t_i}$ est dans ce contexte approximé sur cette grille temporelle, $\mathcal{X}^h$ est une approximation simulable de $\mathcal{X}$ en temps discret (par exemple en utilisant un schéma d'Euler pour X) et $\Phi^h_{\bar{t_i}}$ est une certaine approximation en temps discret calculable de $\Phi_{\bar{t_i}}$. Ce schéma proposé est ensuite implémenté en utilisant des régressions par moindres carrés (voir Chapitre 2) et des régressions quantiles (voir Chapitre 3) en utilisant des approximateurs sous forme de réseaux de neurones. En l'absence d'un benchmark Nested Monte Carlo, nous estimons les erreurs de régression locales L^2 en utilisant l'approche de *twin simulation* introduite au Chapitre 2. Outre la simplicité de la formulation de notre schéma explicite, nous démontrons sur une étude de cas XVA la supériorité de notre approche par rapport aux schémas basés sur des itérations de Picard, tant en termes de vitesse de calcul que de stabilité vis-à-vis de la taille des pas de temps.

2.5 Chapitre 5 – Fast Calibration using Complex-Step Sobolev Training

Étant donné un modèle de valorisation paramétré, sa calibration consiste généralement à résoudre un problème de minimisation de la forme:

$$\theta^\star \in \underset{\theta\in\Theta}{\operatorname{argmin}} \sum_{l=1}^{L} \left(p_{\text{model}}(\theta, k^{(l)}, \tau^{(l)}) - p^{(l)}_{\text{mkt}} \right)^2$$

où les instruments de calibration sont des calls européens vanilles, $p_{\text{model}}(\theta, k, \tau)$ est le prix d'un call vanille avec un prix d'exercice k et une maturité τ dans notre modèle de valorisation paramétré par $\theta \in \Theta$, $\Theta \subset \mathbb{R}^n$ est un ensemble de paramètres de modèle admissibles. Nous avons également accès à L prix de marché de calls vanilles $p^{(1)}_{\text{mkt}}, \ldots, p^{(L)}_{\text{mkt}}$ correspondant aux prix d'exercice $k^{(1)}, \ldots, k^{(L)}$ et maturités $\tau^{(1)}, \ldots, \tau^{(L)}$.

Sauf dans des contextes spéciaux où la minimisation ci-dessus peut être résolue sous forme fermée (*e.g.* dans un modèle de volatilité locale utilisant la formule de Dupire), il faut en général résoudre le problème en utilisant une routine d'optimisation numérique qui implique souvent des évaluations répétées de la fonction objectif[11]. Le coût en temps de calcul de la calibration peut parfois être prohibitif si p_{model} est difficile à calculer, c'est-à-dire si on ne dispose pas d'une formule analytique ou semi-analytique pour le prix de modèle. C'est le cas par exemple pour les modèles à *volatilité rugueuse* [Bayer et al., 2016].

Pour contourner ce problème, des approches ont été proposées où on construit d'abord une approximation rapide de p_{model} en fonction des paramètres du modèle et du produit. Celles-ci utilisent des techniques d'apprentissage automatique, comme celles basées sur des réseaux de neurones [Horvath et al., 2021; Bayer and Stemper, 2018] ou des régressions de processus gaussiens [De Spiegeleer et al., 2018]. Ces approches apprennent une approximation de la fonction de prix de modèle à l'aide d'ensembles de données préconstruits constitués de combinaisons de paramètres de modèle/produit et de prix de modèle obtenus à l'aide d'une routine Monte Carlo, traitant ainsi la fonction de prix de modèle comme une boîte noire.

Dans ce chapitre, nous proposons de tirer parti du fait que le prix du modèle est une espérance conditionnelle du payoff considéré, noté Z^Ξ, *i.e.*

$$p_{\text{model}}(\Xi, K, T) = \mathbb{E}[Z^\Xi | \Xi, K, T]$$

11. et de son gradient en cas d'utilisation d'un optimiseur d'ordre 1, *i.e.* nécéssitant des gradients.

où Ξ, K et T sont des versions randomisées, dans un sens à préciser, respectivement du vecteur des paramètres du modèle, du prix d'exercice et de la maturité. Nous avons négligé l'actualisation pour simplifier cette introduction. Par conséquent, le prix de modèle correspondant aux paramètres du modèle et du produit Ξ, K et T, vu comme une projection L^2 de Z^Ξ sur le sous-espace vectoriel constitué de variables aléatoires $\sigma(\Xi, K, T)$-mesurables de carrés intégrables, est un minimiseur de l'erreur de projection L^2 associée, c'est-à-dire

$$p_{\text{model}} \in \operatorname*{argmin}_{\varphi \in \mathcal{B}} \mathbb{E}[(\varphi(\Xi, K, T) - Z^\Xi)^2].$$

On pourrait alors effectuer la minimisation sur un espace bien choisi de réseaux de neurones $\mathcal{N}$ comme nous l'avons fait dans les chapitres précédents pour des problèmes de projection similaires, c'est-à-dire chercher une approximation p_{proxy} telle que

$$p_{\text{proxy}} \in \operatorname*{argmin}_{\varphi \in \mathcal{N}} \mathbb{E}[(\varphi(\Xi, K, T) - Z^\Xi)^2].$$

Dans le présent chapitre[12], cependant, nous proposons d'enrichir l'apprentissage en utilisant des dérivées *trajectorielles* du payoff par rapport aux paramètres du modèle et du produit, c'est-à-dire que nous résolvons à la place le problème de minimisation suivant

$$p_{\text{proxy}} \in \operatorname*{argmin}_{\varphi \in \mathcal{N}} \mathbb{E}[(\varphi(\Xi, K, T) - Z^\Xi)^2] + \sum_{k=1}^{n+2} \lambda_k \mathbb{E}[(\partial_k \varphi(\Xi, K, T) - \partial_k Z^\Xi)^2] \tag{7}$$

où $\lambda \in (\mathbb{R}_+^\star)^{n+2}$, les réseaux dans $\mathcal{N}$ sont supposés suffisamment réguliers et ∂_k est la dérivée partielle par rapport à la k-ème composante de la concaténation de Ξ et (K, T). $\partial_k Z^\Xi$ est une dérivée dite *trajectorielle* [Broadie and Glasserman, 1996] que l'on rappelle et pour laquelle on donne suffisamment d'hypothèses pour que l'on ait

$$\partial_k \mathbb{E}[Z^\Xi | \Xi, K, T] = \mathbb{E}[\partial_k Z^\Xi | \Xi, K, T]. \tag{8}$$

Ainsi, à la lumière de (8), dans le problème d'apprentissage exprimé dans (7) on cherche alors explicitement à projeter à la fois le payoff et ses dérivées trajectorielles. Enrichir l'apprentissage avec des informations sur les dérivées a d'abord été[13] étudié dans un contexte de réseaux de neurones dans [Czarnecki et al., 2017] sous le nom d'*entraînement Sobolev*, avec Huge and Savine, 2020 appliquant plus tard la même approche à l'apprentissage des prix vus comme des fonctions des valeurs initiales de l'EDS du sous-jacent. Dans un contexte de minimisation du risque empirique, cette approche est plus efficace que la simulation par force brute de plusieurs trajectoires, car les dérivés trajectorielles peuvent partager non seulement les mêmes nombres aléatoires, mais aussi généralement de nombreuses sous-expressions communes.

Nous montrons également qu'on évite la charge de calcul associée à l'évaluation des erreurs quadratiques pour chaque dérivée partielle dans (7) en calculant à la place une erreur quadratique de projection associée à une seule dérivée directionnelle dans une direction aléatoire. En effet, nous avons:

$$\mathbb{E}[(u^\top \nabla \varphi(\Xi, K, T) - u^\top \nabla (e^{-rT} Z^\Xi))^2] = \sum_{k=1}^{n+2} \lambda_k \mathbb{E}[(\partial_k \varphi(\Xi, K, T) - \partial_k (e^{-rT} Z^\Xi))^2]$$

12. Nous n'avons pas utilisé cette approche, du moins dans sa forme actuelle, dans le cadre XVA des chapitres précédents car elle est non triviale du point de vue de l'occupation de l'espace mémoire. En effet, les dérivées trajectorielles devront être calculées et conservées en mémoire à chaque pas de temps grossier. De plus, cette approche ne s'applique pas directement aux mesures de risque conditionnelles, car on ne peut pas, par exemple, intervertir les opérateurs Value-at-Risk conditionnelle et dérivée.

13. Du moins à notre connaissance.

pour tout vecteur aléatoire L^2-intégrable u supporté sur $\mathbb{R}^{n+2}$ à composantes centrées telles que $\mathrm{cov}(u) = \mathrm{diag}(\lambda_1, \dots, \lambda_{n+2})$ et u est indépendant de Ξ, K, T, Z^{Ξ}. Nous montrons également comment choisir la distribution sur u telle que la variance ajoutée[14] dans l'intégrand de (7) est minimisée.

Nous proposons ensuite de calculer la dérivée directionnelle $u^\top \nabla\varphi(\Xi, K, T)$ avec une erreur proche de la précision machine en utilisant la *différenciation à pas complexe* [Martins et al., 2003; Squire and Trapp, 1998] tout en restant en simple précision. Ceci consiste à introduire une petite perturbation imaginaire dans le sens de la différenciation, c'est-à-dire

$$\frac{1}{\varepsilon}\mathrm{Im}(\varphi(\Xi + \mathrm{i}\,\varepsilon\, u_{1:n}, K + \mathrm{i}\,\varepsilon\, u_{n+1}, T + \mathrm{i}\,\varepsilon\, u_{n+2})) = u^\top \nabla\varphi(\Xi, K, T) + \mathcal{O}(\varepsilon^2)$$

lorsque $\varepsilon \to 0$, pour tout φ analytique et au moins trois fois différentiable par rapport à ses entrées et où i est l'unité imaginaire. En revanche, la méthode des différences finies échoue généralement pour des tailles de pas trop petites en raison d'erreurs d'arrondi en simple précision, sauf si on passe à la double précision, ce qui nuirait aux performances sur GPU et utiliserait deux fois la quantité de mémoire nécessaire en simple précision.

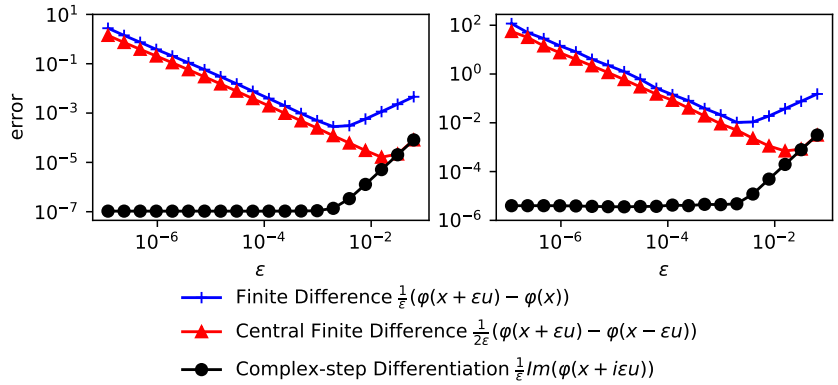

Figure 10. Moyenne empirique de la valeur absolue des erreurs absolues (**gauche**) et relatives (**droite**), lors de l'approximation de la dérivée directionnelle d'un réseau de neurones initialisé aléatoirement φ par rapport à ses entrées, en utilisant chacune des méthodes de différence finie, de différence finie centrale et de différenciation à pas complexe. Ce réseau comporte 28 entrées, 6 couches cachées, 112 unités cachées par couche et une activation analytique Softplus. La moyenne est faite sur un échantillon i.i.d de $2^{14} = 16384$ erreurs correspondant chacune à un vecteur d'entrée x avec des composantes tirées indépendamment l'une des autres et de loi $\mathcal{U}([-\sqrt{3}, \sqrt{3}])$ et une direction u tirée comme dans la Proposition 5.10 avec $\lambda_1 = \cdots = \lambda_{28} = 1$. Les graphes sont dans une échelle log-log.

Nous montrons que cette approche pour calculer les dérivées directionnelles randomisées est plus rapide qu'un calcul exact utilisant des produits vecteur-jacobien dans la direction *forward*. Nous appelons l'approche d'apprentissage qui en résulte *entraînement Sobolev à pas complexe*.

Nous montrons l'efficacité de notre méthode en la testant numériquement sur un exemple de modèle de volatilité locale à grille fixe 5×5 où chaque nœud de volatilité locale est traité comme un paramètre de modèle, donnant donc lieu à 25 paramètres de modèle au total. Nous fournissons également des benchmarks présentant les accélérations obtenues en utilisant notre approche. Nous montrons que cette approche est statistiquement plus efficace que la méthode *brute force* qui consiste à générer plus de chemins de Monte Carlo.

14. En effet, comme nous le montrons dans le chapitre, randomiser la direction de différenciation augmente nécessairement la variance de l'intégrand dans (7).

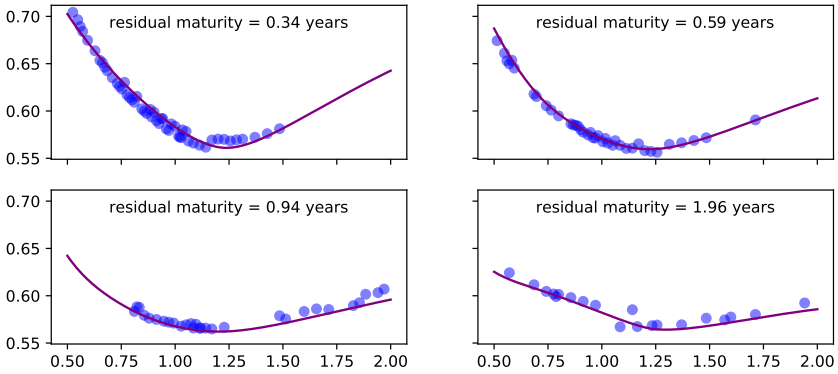

Figure 11. Ajustement du smile de volatilité implicite TSLA au 2022/02/14 en utilisant un modèle de volatilité locale 5×5 calibré à l'aide de notre approximation de prix. **Points en bleu:** prix du marché, **Courbe en violet:** smile de volatilité implicite du modèle de volatilité locale calibré, **abscisses:** moneyness, **ordonnées:** niveaux de volatilité implicite.

Chapter 1

XVA Analysis From the Balance Sheet

This chapter, published as a paper in [Albanese et al., 2021], was co-authored with Claudio Albanese, Stéphane Crépey and Rodney Hoskinson.

XVAs denote various counterparty risk related valuation adjustments that are applied to financial derivatives since the 2007–09 crisis. We root a cost-of-capital XVA strategy in a balance sheet perspective which is key in identifying the economic meaning of the XVA terms. Our approach is first detailed in a static setup that is solved explicitly. It is then plugged in the dynamic and trade incremental context of a real derivative banking portfolio. The corresponding cost-of-capital XVA strategy ensures to bank shareholders a submartingale equity process corresponding to a target hurdle rate on their capital at risk, consistently between and throughout deals. Set on a forward/backward SDE formulation, this strategy can be solved efficiently using GPU computing combined with deep learning regression methods in a whole bank balance sheet context. A numerical case study emphasizes the workability and added value of the ensuing pathwise XVA computations.

1.1 Introduction

XVAs, with X as C for credit, D for debt, F for funding, M for margin, or K for capital, are post-2007–09 crisis valuation adjustments for financial derivatives. In broad terms to be detailed later in the paper (cf. Table 1.1 in Section 1.2), CVA is what the bank expects to lose due to counterparty defaults in the future; DVA (irrelevant for pricing but material to bank creditors as we will see) is what the bank expects to gain due to its own default; FVA is the expected cost for the bank of having to raise variation margin (re-hypothecable collateral); MVA is the expected cost for the bank of having to raise initial margin (segregated collateral); KVA is the expected cost for the bank of having to remunerate its shareholders through dividends for their capital at risk.

XVAs deeply affect the derivative pricing task by making it global, nonlinear, and entity dependent. However, before these technical implications, the fundamental point is to understand what really deserves to be priced and what does not, by rooting the pricing approach in a corresponding collateralization, accounting, and dividend policy of the bank.

Coming after several papers on the valuation of defaultable assets in the 90's, such as [Duffie and Huang, 1996], [Bielecki and Rutkowski, 2002] (Eq. (14.25) p. 448) obtained the formula

$$\text{CVA} - \text{DVA} \tag{1.1}$$

for the valuation of bilateral counterparty risk on a swap, assuming risk-free funding. This formula, rediscovered and generalized by others since the 2008–09 financial crisis (cf. e.g. [Brigo and Capponi, 2010]), is symmetrical, i.e. it is the negative of the analogous quantity considered from the point of view of the counterparty, consistent with the law of one price and the Modigliani and Miller, 1958 theorem.

Around 2010, the materiality of the DVA windfall benefit of a bank at its own default time became the topic of intense debates in the quant and academic communities. At least, it seemed reasonable to admit that, if the own default risk of the bank was accounted for in the modeling, in the form of a DVA benefit, then the cost of funding (FVA) implication of this risk should be included as well, leading to the modified formula (CVA − DVA + FVA). See for instance [Burgard and Kjaer, 2011, 2013, 2017], [Crépey, 2015], [Brigo and Pallavicini, 2014], or [Bichuch et al., 2018]. See also [Bielecki and Rutkowski, 2015] for an abstract funding framework (without explicit reference to XVAs), generalizing [Piterbarg, 2010] to a nonlinear setup.

Then Hull and White, 2012 objected that the FVA was only the compensator of another windfall benefit of the bank at its own default, corresponding to the non-reimbursement by the bank of its funding debt. Accounting for the corresponding "DVA2" (akin to the FDA in this paper) brings back to the original firm valuation formula:

$$\text{CVA} - \text{DVA} + \text{FVA} - \text{FDA} = \text{CVA} - \text{DVA},$$

as FVA = FDA (assuming risky funding fairly priced as we will see).

However, their argument implicitly assumes that the bank can perfectly hedge its own default: cf. [Burgard and Kjaer, 2013](end of Section 3.1) and see Section 1.3.5 below. As a bank is an intrinsically leveraged entity, this is not the case in practice. One can mention the related corporate finance notion of debt overhang in Myers, 1977, by which a project valuable for the firm as a whole may be rejected by shareholders because the project is mainly valuable to bondholders. But, until recently, such considerations were hardly considered in the field of derivative pricing.

The first ones to recast the XVA debate in the perspective of the balance sheet of the bank were Burgard and Kjaer, 2011, to explain that an appropriately hedged derivative position has no impact on the dealer's funding costs. Also relying on balance sheet models of a dealer bank, Castagna, 2014 and Andersen et al., 2019 end up with conflicting conclusions, namely that the FVA should, respectively should not, be included in the valuation of financial derivatives. Adding the KVA, but in a replication framework, Green et al., 2014 conclude that both the FVA and the KVA should be included as add-ons in entry prices and as liabilities in the balance sheet.

1.1.1 Contents

Our key premise is that counterparty risk entails two distinct but intertwined sources of

market incompleteness:

- A bank cannot perfectly hedge counterparty default losses, by lack of sufficiently liquid CDS markets;

- A bank can even less hedge its own jump-to-default exposure, because this would mean selling protection on its own default, which is nonpractical and, under certain juridictions, even legally forbidden (see Section 1.2).

We specify the banking XVA metrics that align derivative entry prices to shareholder interest, given this impossibility for a bank to replicate the jump-to-default related cash flows. We develop a cost-of-capital XVA approach consistent with the accounting standards set out in IFRS 4 Phase II (see International Financial Reporting Standards, 2013), inspired from the Swiss solvency test and Solvency II insurance regulatory frameworks (see Swiss Federal Office of Private Insurance, 2006 and Committee of European Insurance and Occupational Pensions Supervisors, 2010), which so far has no analogue in the banking domain. Under this approach, the valuation (CL) of the so-called contra-liabilities and the cost of capital (KVA) are sourced from clients at trade inceptions, on top of the (CVA − DVA) complete market valuation of counterparty risk, in order to compensate bank shareholders for wealth transfer and risk on their capital.

The cost of the corresponding collateralization, accounting, and dividend policy is, by contrast with the complete market valuation (CVA − DVA) of counterparty risk,

$$\mathrm{CVA} + \mathrm{FVA} + \mathrm{KVA}, \tag{1.2}$$

computed unilaterally in a certain sense (even though we do crucially include the default of the bank itself in our modeling), and charged to clients on an incremental run-off basis at every new deal[1.1].

All in one, our cost-of-capital XVA strategy makes shareholder equity a submartingale with drift corresponding to a hurdle rate h on shareholder capital at risk, consistently between and throughout deals. Thus we arrive at a sustainable strategy for profits retention, much like in the above-mentioned insurance regulation, but in a consistent continuous-time and banking framework.

Last but not least, our approach can be solved efficiently using GPU computing combined with deep learning regression methods in a whole bank balance sheet context.

1.1.2 Outline and Contributions

Section 1.2 sets a financial stage where a bank is split across several trading desks and entails different stakeholders. Section 1.3 develops our cost-of-capital XVA approach in a one-period static setup. Section 1.4 revisits the approach at the dynamic and trade incremental level. Section 1.5 is a numerical case study on large, multi-counterparty portfolios of interest rate swaps, based on the continuous-time XVA equations for bilateral trade portfolios recalled in Section 1.6.

1.1. See also Remark 1.1 regarding the meaning of the FVA in (1.2).

The ***main contributions*** of the paper are:

- The one-period static XVA model of Section 1.3, with explicit formulas for all the quantities at hand, offering a concrete grasp on the related wealth transfer and risk premium issues;
- Proposition 1.20, which establishes the connections between XVAs and the core equity tier 1 capital of the bank, respectively bank shareholder equity;
- Proposition 1.25, which establishes that, under the XVA policy represented by the balance conditions (1.3) between deals and the counterparty risk add-on (1.50) throughout deals, bank shareholder equity is a submartingale with drift corresponding to a target hurdle rate h on shareholder capital at risk. This perspective solves the puzzle according to which, on the one hand, XVA computations are performed on a run-off portfolio basis, while, on the other hand, they are used for computing pricing add-ons to new deals;
- The XVA deep learning (quantile) regression computational strategy of Section 1.4.4;
- The numerical case study of Section 1.5, which emphasizes the materiality of refined, pathwise XVA computations, as compared to more simplistic XVA approaches.

From a broader point of view, this paper reflects a shift of paradigm regarding the pricing and risk management of financial derivatives, from hedging to balance sheet optimization, as quantified by relevant XVA metrics. In particular (compare with the last paragraph before Section 1.1.1), our approach implies that the FVA (and also the MVA, see Remark 1.1) should be included as an add-on in entry prices and as a liability in the balance sheet; the KVA should be included as an add-on in entry prices, but not as a liability in the balance sheet.

From a computational point of view, this paper opens the way to second generation XVA GPU implementation. The first generation consisted of nested Monte Carlo implemented by explicit CUDA programming on GPUs (see Albanese et al., 2017, Abbas-Turki et al., 2018). The second generation takes advantage of GPUs leveraging via pre-coded CUDA/AAD deep learning packages that are used for the XVA embedded regression and quantile regression task. Compared to a regulatory capital based KVA approach, an economic capital based KVA approach is then not only conceptually more satisfying but also simpler to implement.

1.2 Balance Sheet and Capital Structure Model of the Bank

We consider a dealer bank, which is a market maker involved in bilateral derivative portfolios. For simplicity, we only consider European derivatives. The bank has two kinds of stakeholders, ***shareholders*** and ***bondholders***. The shareholders have the control of the bank and are solely responsible for investment decisions before bank default. The bondholders represent the senior creditors of the bank, who have no decision power until bank default, but are protected by laws, of the pari-passu type, forbidding trades that would trigger value away from them to shareholders during the default resolution process of

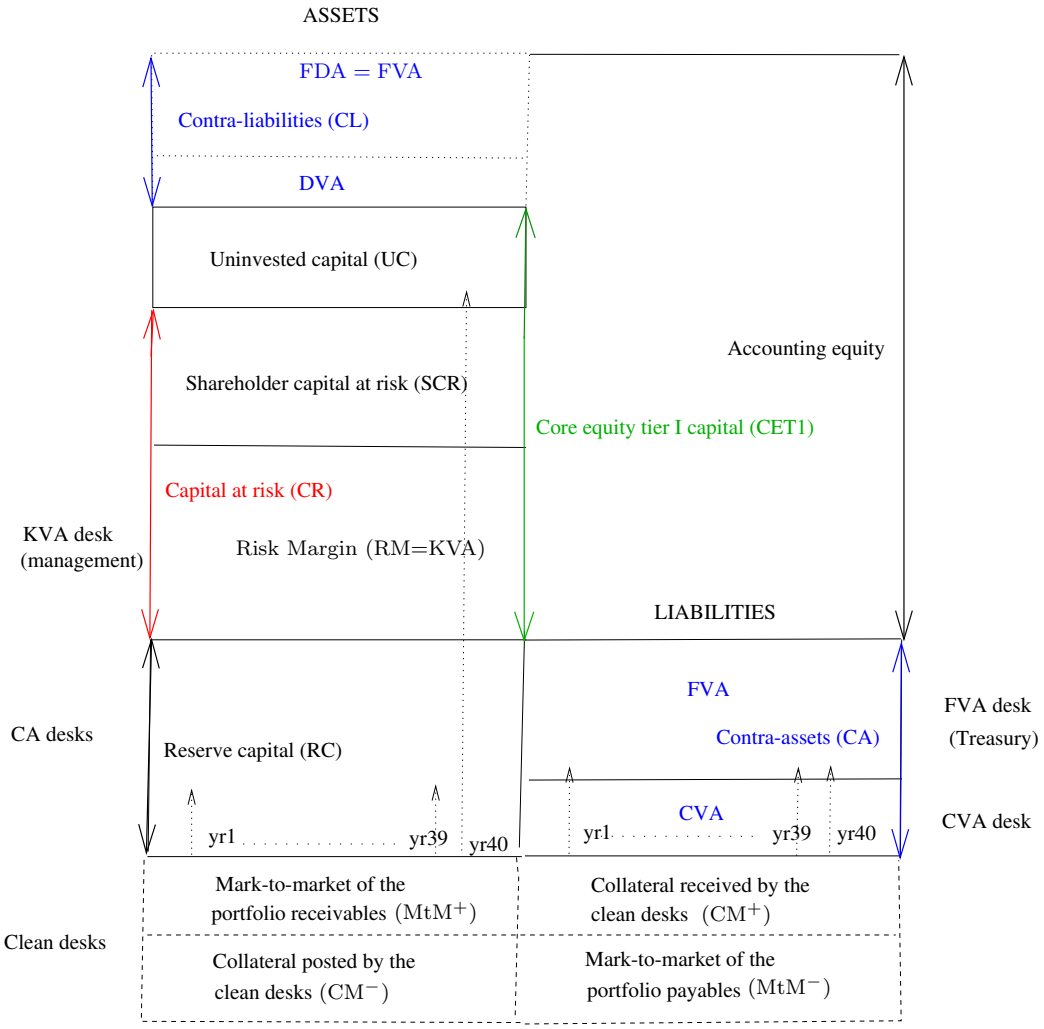

Figure 1.1. Balance sheet of a dealer bank. Contra-liability valuation (CL) at the top is shown in dotted boxes because it is only value to the bondholders (see Section 1.3.5). Mark-to-market valuation (MtM) of the derivative portfolio of the bank by the clean desks, as well as the corresponding collateral (clean margin CM), are shown in dashed boxes at the bottom. Their role will essentially vanish in our setup, where we assume a perfect clean hedge by the bank. The arrows in the left column represent trading losses of the CA desks in "normal years 1 to 39" and in an "exceptional year 40" with full depletion (i.e. refill via UC, under Assumption 1.5.ii) of RC, RM, and SCR. The numberings yr1 to yr40 are fictitious yearly scenarios in line with a 97.5% expected shortfall of the one-year-ahead trading losses of the bank that we use for defining its economic capital. The arrows in the right column symbolize the average depreciation in time of contra-assets between deals. The collateral between the bank and its counterparties is not shown to alleviate the picture.

the bank. The bank also has junior creditors, represented in our framework by an ***external funder***, who can lend unsecured to the bank and is assumed to suffer an exogenously given loss-given-default in case of default of the bank.

We consider three kinds of business units within the bank (see Figure 1.1 for the corresponding picture of the bank balance sheet and refer to Table 1.1 for a list of the main financial acronyms used in the paper): the ***CA desks***, i.e. the CVA desk and the FVA desk (or Treasury) of the bank, in charge of contra-assets, i.e. of counterparty risk and its funding implications for the bank; the ***clean desks***, who focus on the market risk of the contracts in their respective business lines; the ***management*** of the bank, in charge of the dividend release policy of the bank.

Amounts on dedicated cash accounts of the bank:

CM	Clean margin	Definition 1.2 and Assumption 1.5
RC	Reserve capital	Definition 1.2 and Assumption 1.5
RM	Risk margin	Definition 1.2 and Assumption 1.5
UC	Uninvested capital	Definition 1.2 and Assumption 1.5

Valuations:

CA	Contra-assets valuation	(1.2), (1.16), and (1.64)
CL	Contra-liabilities valuation	Definition 1.2 and (1.19), (1.41), and (1.50)
CVA	Credit valuation adjustment	(1.17), (1.16), (1.65), and (1.76)–(1.77)
DVA	Debt valuation adjustment	(1.19) and (1.17)
FDA	Funding debt adjustment	(1.19) and (1.26)
FV	Firm valuation of counterparty risk	(1.22) and (1.26)
FVA	Funding valuation adjustment	Remark 1.1, (1.17), (1.16), and (1.65)
KVA	Capital valuation adjustment	(1.3), (1.31), and (1.71)
MtM	Mark-to-market	(1.3) and (1.15)
MVA	Margin valuation adjustment	Remark 1.1, (1.37), (1.65), and (1.78)
XVA	Generic "X" valuation adjustment	First paragraph

Also:

CR	Capital at risk	(1.69)
CET1	Core equity tier I capital	(1.2) and (1.47)
EC	Economic capital	Definitions 1.13 and 1.27
FTP	Funds transfer price	(1.50)
SHC	Shareholder capital (or equity)	(1.2) and (1.48)
SCR	Shareholder capital at risk	Assumption 1.5 and (1.30)

Table 1.1. Main financial acronyms and place where they are introduced conceptually and/or specified mathematically in the paper, as relevant.

Collateral means cash or liquid assets that are posted to guarantee a netted set of transactions against defaults. It comes in two forms: variation margin, which is re-hypothecableotecable, i.e. fungible across netting sets, and initial margin, which is segregated. We assume cash only collateral. Posted collateral is supposed to be remunerated at the risk-free rate (assumed to exist, with overnight index swap rates as a best market proxy).

Remark 1.1. To alleviate the notation, in this conceptual section of the paper, we only consider an FVA as the global cost of raising collateral for the bank, as opposed to a distinction, in the industry and in later sections in the paper, between an FVA, in the strict sense of the cost of raising variation margin, and an MVA for the cost of raising initial margin. □

The CA desks guarantee the trading of the clean desks against counterparty defaults, through a ***clean margin account***, which can be seen as (re-hypothecable) collateral exchanged between the CA desks and the clean desks. The corresponding clean margin amount (CM) also plays the role of the funding debt of the clean desks put at their disposal at a risk-free cost by the Treasury of the bank. This is at least the case when $\mathrm{CM} > 0$ (clean desks clean margin receivers). In the case when $\mathrm{CM} < 0$ (clean desks clean margin posters), $(-\mathrm{CM})$ corresponds to excess cash generated by the trading of the clean desks, usable by the Treasury for its other funding purposes. See the bottom, dashed boxes in Figure 1.1.

In addition, the CA desks value the contra-assets (future counterparty default losses and funding expenditures), charge them to the (corporate) clients at deal inception, deposit the corresponding payments in a ***reserve capital account***, and then are exposed to the corresponding payoffs. As time proceeds, contra-assets realize and are covered by the CA desks with the reserve capital account.

On top of reserve capital, the so-called risk margin is sourced by the management of the bank from the clients at deal inception, deposited into a ***risk margin account***, and then gradually released as KVA payments into the shareholder dividend stream.

Another account contains the ***shareholder capital at risk*** earmarked by the bank to deal with exceptional trading losses (beyond the expected losses that are already accounted for by reserve capital).

Last, there is one more bank account with shareholder ***uninvested capital***.

All cash accounts are remunerated at the risk-free rate.

Definition 1.2. *We write CM, RC, RM, SCR, and UC for the respective (risk-free discounted) amounts on the clean margin, reserve capital, risk margin, shareholder capital at risk, and uninvested capital accounts of the bank. We also define*

$$\mathrm{SHC} = \mathrm{SCR} + \mathrm{UC}, \quad \mathrm{CET1} = \mathrm{RM} + \mathrm{SCR} + \mathrm{UC}. \square$$

From a financial interpretation point of view, before bank default, SHC corresponds to shareholder capital (or equity); CET1 is the ***core equity tier I capital*** of the bank, representing the financial strength of the bank assessed from a regulatory, structural solvency point of view, i.e. the sum between shareholder capital and the risk margin (which is also loss-absorbing), but excluding the value CL of the so-called contra-liabilities (see Figure 1.1). Indeed, the latter only benefits the bondholders (cf. Section 1.3.5), hence it only enters accounting equity. Before the default of the bank, ***shareholder wealth*** and ***bondholder wealth*** are respectively given by $\mathrm{SHC} + \mathrm{RM}^{sh}$ and $\mathrm{CL} + \mathrm{RM}^{bh}$, for shareholder and bondholder components of RM to be detailed in Remark 1.15; shareholder and bondholder wealths sum up to the accounting equity $\mathrm{RM} + \mathrm{SCR} + \mathrm{UC} + \mathrm{CL}$, i.e. the wealth of the firm as a whole (see Figure 1.1).

Remark 1.3. The purpose of our capital structure model of the bank is *not* to model the default of the bank, like in a Merton, 1974 model, as the point of negative equity (i.e. $\mathrm{CET} < 0$). In the case of a bank, such a default model would be unrealistic. For instance, at the time of its collapse in April 2008, Bear Stearns had billions of capital. In fact, the legal definition of default is an unpaid coupon or cash flow, which is a liquidity (as opposed to solvency) issue. Eventually we will model the default of the bank as a totally unpredictable event at some exogenous time τ calibrated to the credit default swap (CDS) curve referencing the bank. Indeed we view the latter as the most reliable and informative credit data regarding anticipations of markets participants about future recapitalization, government intervention, bail-in, and other bank failure resolution policies.

The aim of our capital structure model, instead, is to put in a balance sheet perspective the contra-assets and contra-liabilities of a dealer bank, items which are not present in the Merton model and play a key role in our XVA analysis. $\square$

In line with the Volcker rule banning proprietary trading for a bank, we assume a perfect market hedge of the derivative portfolio of the bank by the clean desks, in a sense to be specified below in the respective static and continuous-time setups. By contrast, as jump-to-default exposures (own jump-to-default exposure, in particular) cannot be hedged by the bank (cf. Section 1.1.1), we conservatively assume no XVA hedge.

We work on a measurable space $(\Omega, \mathcal{A})$ endowed with a probability measure $\mathbb{Q}^\star$, with $\mathbb{Q}^\star$ expectation denoted by $\mathbb{E}^\star$, which is used for the linear valuation task, using the risk-free asset as our numéraire everywhere.

Remark 1.4. Regarding the nature of our reference probability measure $\mathbb{Q}$, "physical or risk-neutral", one should view it as a blend between the two. For instance, even if we do not use this explicitly in the paper, one could conceptually think of $\mathbb{Q}^\star$ as the probability measure introduced by Dybvig, 1992 to deal with incomplete markets that are a mix of financial traded risk factors and unhedgeable ones (jumps to default, in our setup), recently revisited in a finance and insurance context by Artzner et al., 2020. Namely, one could think of Q as the unique probability measure on $\mathcal{A}$[1.2] that coincides *(i)* with a given risk-neutral pricing measure *on* the financial σ algebra $\subseteq A$, and *(ii)* with the physical probability measure *conditional* on the financial σ algebra (the risk-neutral and physical measures being assumed equivalent on the financial σ algebra). The risk-neutral pricing measure (hence, in view of *(i)*, $\mathbb{Q}^\star$ itself) is calibrated to prices of fully collateralized transaction for which counterparty risk is immaterial. The physical probability measure expresses user views on the unhedgeable risk factors. The uncertainty about $\mathbb{Q}^\star$ can be dealt with by a Bayesian variation on our baseline XVA approach, whereby paths of alternative, co-calibrated models are combined in a global simulation (cf. Hoeting et al., 1999). □

1.2.1 Run-Off Portfolio

Until Section 1.4.2, we consider the case of a portfolio held on a run-off basis, i.e. set up at time 0 and such that no new unplanned trades enter the portfolio in the future.

The trading cash flows of the bank (cumulative cash flow streams starting from 0 at time 0) then consist of

- the contractually promised cash flows $\mathcal{P}$ from counterparties,
- the counterparty credit cash flows $\mathcal{C}$ to counterparties (i.e., because of counterparty risk, the effective cash flows from counterparties are $\mathcal{P} - \mathcal{C}$),
- the risky funding cash flows $\mathcal{F}$ to the external funder, and
- the hedging cash flows $\mathcal{H}$ of the clean desks to financial hedging markets

(note that all cash flow differentials can be positive or negative). See Section 1.3.1 and (1.58)–(1.61) for concrete specifications in respective one-period and continuous-time setups.

Assumption 1.5.

1. ***(Self-financing condition)*** $\mathrm{RC} + \mathrm{RM} + \mathrm{SCR} + \mathrm{UC} - \mathrm{CM}$ *evolves like the received trading cash flows* $\mathcal{P} - \mathcal{C} - \mathcal{F} - \mathcal{H}$.
2. ***(Mark-to-model)*** *The amounts on all the accounts but UC are marked-to-model (hence the last, residual amount, UC, plays the role of an adjustment variable). Specifically, we assume that the following* ***shareholder balance conditions*** *hold at all times:*

$$\mathrm{CM} = \mathrm{MtM}, \quad \mathrm{RC} = \mathrm{CA}, \quad \mathrm{RM} = \mathrm{KVA}, \tag{1.3}$$

   for theoretical target levels MtM, CA, *and* KVA *to be specified in later sections of the paper (which will also determine the theoretical target level for* SCR*).*
3. ***(Agents)*** *The initial amounts* MtM_0, CA_0, *and* KVA_0 *are provided by the clients at portfolio inception time 0. Resets between time 0 and the bank default time* τ *(excluded) are on bank shareholders. At the (positive) bank default time* τ*, the property of the residual amount on the reserve capital and risk margin accounts is transferred from the shareholders to the bondholders of the bank.* □

1.2. See [Artzner et al., 2020] (Proposition 2.1) for a proof.

Remark 1.6. In an asymmetric setup with a price maker and a price taker, the price maker passes his costs to the price taker. Accordingly, in our setup, the (corporate) clients provide all the amounts to the clean margin, reserve capital, and risk margin accounts of the bank required for resetting the accounts to their theoretical target levels (1.3) corresponding to the updated portfolio. □

Under a cost-of-capital XVA approach, we define valuation so as to make shareholder trading losses (that include marked-to-model liability fluctuations) centered, then we add a KVA risk premium in order to ensure to bank shareholders some positive hurdle rate h on their capital at risk.

In what follows, such an approach is developed, first, in a static setup, which can be solved explicitly, and then, in a dynamic and trade incremental setup, as suitable for dealing with a real derivative banking portfolio.

1.3 XVA Analysis in a Static Setup

In this section, we apply the cost-of-capital XVA approach to a portfolio made of a single deal, $\mathcal{P}$ (random variable promised to the bank), between a bank and a client, without prior endowment, in an elementary one-period (one year) setup. All the trading cash flows $\mathcal{P}$, $\mathcal{C}$, $\mathcal{F}$, and $\mathcal{H}$ are then random variables (as opposed to processes in a multi-period setup later in the paper). We first assume no collateral exchanged between the bank and its client (but collateral exchanged as always between the CA and the clean desks as well as collateral on the market hedge of the bank, the way explained after the respective Remarks 1.1 and 1.3). Risky funding assets are assumed fairly priced by the market, in the sense that $\mathbb{E}^\star\mathcal{F} = 0$.

The bank and client are both default prone with zero recovery to each other. The bank also has zero recovery to its external funder. We denote by J and J_1 the survival indicators (random variables) of the bank and client at time 1, with default probability of the bank $\mathbb{Q}^\star(J = 0) = \gamma$.

Since prices and XVAs only matter at time 0 in a one-period setup, we identify all the XVA processes, as well as the mark-to-market (valuation by the clean desks) MtM of the deal, with their values at time 0.

For any random variable $\mathcal{Y}$, we define

$$\mathcal{Y}^\circ = J\mathcal{Y} \text{ and } \mathcal{Y}^\bullet = -(1-J)\mathcal{Y}, \text{ hence } \mathcal{Y} = \mathcal{Y}^\circ - \mathcal{Y}^\bullet.$$

Let $\mathbb{E}$ denote the expectation with respect to the bank survival measure, say $\mathbb{Q}$, associated with $\mathbb{Q}^\star$, i.e., for any random variable $\mathcal{Y}$,

$$\mathbb{E}\mathcal{Y} = (1-\gamma)^{-1}\mathbb{E}^\star(\mathcal{Y}^\circ) \tag{1.4}$$

(which is also equal to $\mathbb{E}\mathcal{Y}^\circ$). The notion of bank survival measure was introduced in greater generality by Schönbucher, 2004. In the present static setup, (1.4) is nothing but the $\mathbb{Q}^\star$ expectation of $\mathcal{Y}$ conditional on the survival of the bank (note that, whenever $\mathcal{Y}$ is independent from J, the right-hand-side in (1.4) coincides with $\mathbb{E}^\star\mathcal{Y}$).

Lemma 1.7. *For any random variable $\mathcal{Y}$ and constant Y, we have*

$$Y = \mathbb{E}^\star(\mathcal{Y}^\circ + (1-J)\,Y) \Longleftrightarrow Y = \mathbb{E}\mathcal{Y}.$$

Proof. Indeed,

$$Y = \mathbb{E}^\star(J\mathcal{Y} + (1-J)\,Y) \Longleftrightarrow \mathbb{E}^\star(J\,(\mathcal{Y} - Y)) = 0 \Longleftrightarrow \mathbb{E}(\mathcal{Y} - Y) = 0 \Longleftrightarrow Y = \mathbb{E}\mathcal{Y},$$

where the equivalence in the middle is justified by (1.4). □

Remark 1.8. For simplicity in a first stage, we will ignore the possibility of using capital at risk for funding purposes, only considering in this respect reserve capital $\mathrm{RC} = \mathrm{CA}$ (cf. (1.3)). The additional free funding source provided by capital at risk will be introduced later, as well as collateral between bank and client, in Section 1.3.4.

1.3.1 Cash Flows

Lemma 1.9. *Given the (to be specified)* MtM *and* CA *amounts (cf. Assumption 1.5.ii), the credit and funding cash flows $\mathcal{C}$ and $\mathcal{F}$ of the bank and its trading loss (and profit) L are such that*

$$\mathcal{C}^\circ = J\,(1-J_1)\mathcal{P}^+ = J\gamma\,(\mathrm{MtM}-\mathrm{CA})^+ \tag{1.5}$$

$$\mathcal{C}^\bullet = (1-J)\,(\mathcal{P}^- - (1-J_1)\mathcal{P}^+), \mathcal{F}^\bullet = (1-J)\,((\mathrm{MtM}-\mathrm{CA})^+ - \gamma\,(\mathrm{MtM}-\mathrm{CA})^+) \tag{1.6}$$

$$L^\circ = \mathcal{C}^\circ + \mathcal{F}^\circ - J\,\mathrm{CA}, L^\bullet = \mathcal{C}^\bullet + \mathcal{F}^\bullet + (1-J)\,\mathrm{CA}, L = \mathcal{C} + \mathcal{F} - \mathrm{CA}. \tag{1.7}$$

Proof. For the deal to occur, the bank needs to borrow $(\mathrm{MtM}-\mathrm{CA})^+$ unsecured or invest $(\mathrm{MtM}-\mathrm{CA})^-$ risk-free (cf. Remark 1.8). Having assumed zero recovery to the external funder, unsecured borrowing is fairly priced as $\gamma \times$ the amount borrowed by the bank (in line with our assumption that $\mathbb{E}^\star\mathcal{F} = 0$), i.e. the bank must pay for its risky funding the amount

$$\gamma\,(\mathrm{MtM}-\mathrm{CA})^+.$$

Moreover, at time 1, under zero recovery upon defaults:

- If the bank is not in default (i.e. $J = 1$), then the bank closes its position with the client while receiving $\mathcal{P}$ from its client if the latter is not in default (i.e. $J_1 = 1$), whereas the bank pays $\mathcal{P}^-$ to its client if the latter is in default (i.e. $J_1 = 0$). In addition, the bank reimburses its funding debt $(\mathrm{MtM}-\mathrm{CA})^+$ or receives back the amount $(\mathrm{MtM}-\mathrm{CA})^-$ it had lent at time 0;
- If the bank is in default (i.e. $J = 0$), then the bank receives back $J_1\,\mathcal{P}^+$ on the derivative as well as the amount $(\mathrm{MtM}-\mathrm{CA})^-$ it had lent at time 0.

Also accounting for the hedging loss $\mathcal{H}$, the trading loss of the bank over the year is

$$L = \gamma\,(\mathrm{MtM}-\mathrm{CA})^+ - J\,(J_1\,\mathcal{P} - (1-J_1)\mathcal{P}^- - (\mathrm{MtM}-\mathrm{CA})^+ + (\mathrm{MtM}-\mathrm{CA})^-) \tag{1.8}$$

$$- (1-J)\,(J_1\,\mathcal{P}^+ + (\mathrm{MtM}-\mathrm{CA})^-) + \mathcal{H}. \tag{1.9}$$

In the static setup, the perfect clean hedge condition (see after Remark 1.3) writes $\mathcal{H} = \mathcal{P} - \mathrm{MtM}$. Inserting this into the above yields

$$L = (1-J_1)\,\mathcal{P}^+ + \gamma\,(\mathrm{MtM}-\mathrm{CA})^+ - \mathrm{CA} - (1-J)\,(\mathcal{P}^- + (\mathrm{MtM}-\mathrm{CA})^+), \tag{1.10}$$

as easily checked for each of the four possible values of the pair (J, J_1). That is,

$$L^\circ = \underbrace{J\,(1-J_1)\,\mathcal{P}^+}_{\mathcal{C}^\circ} + \underbrace{J\gamma\,(\mathrm{MtM}-\mathrm{CA})^+}_{\mathcal{F}^\circ} - J\,\mathrm{CA} \tag{1.11}$$

$$L^\bullet = \underbrace{(1-J)\,(\mathcal{P}^- - (1-J_1)\,\mathcal{P}^+)}_{\mathcal{C}^\bullet} + \underbrace{(1-J)\,((\mathrm{MtM}-\mathrm{CA})^+ - \gamma\,(\mathrm{MtM}-\mathrm{CA})^+)}_{\mathcal{F}^\bullet} + (1-J)\,\mathrm{CA}, \tag{1.12}$$

where the identification of the different terms as part of $\mathcal{C}$ or $\mathcal{F}$ follows from their financial interpretation. □

Remark 1.10. The derivation (1.8) implicitly allows for negative equity (that arises whenever $L^\circ > \mathrm{CET1}$, cf. (1.2)), which is interpreted as recapitalization. In a variant of the model excluding both recapitalization and negative equity, the default of the bank would be modeled in a structural fashion as the event $\{L = \mathrm{CET1}\}$, where

$$L = ((1 - J_1)\,\mathcal{P}^+ + \gamma\,(\mathrm{MtM} - \mathrm{CA})^+ - \mathrm{CA}) \wedge \mathrm{CET1}, \tag{1.13}$$

and we would obtain, instead of (1.10), the following trading loss for the bank:

$$\mathbb{1}_{\{\mathrm{CET1} > L\}}\, L + \mathbb{1}_{\{\mathrm{CET1} = L\}}(\mathrm{CET1} - \mathcal{P}^- - (\mathrm{MtM} - \mathrm{CA})^+). \tag{1.14}$$

In this paper we consider a model with recapitalization for the reasons explained in Remark 1.3.

Structural XVA approaches in a static setup have been proposed in Andersen et al., 2019 (without KVA) and Kjaer, 2019 (including the KVA). Their marginal, limiting results as a new deal size goes to zero are comparable to some of the results that we have here. But then, instead of developing a continuous time version of their corporate finance model and taking the small trade limit, these papers start the development of the continuous time model from the single period small trade limit model. By contrast, in our framework, we have end to end development in the continuous time model of Section 1.4 and in the present single period model.

1.3.2 Contra-assets and Contra-liabilities

To make shareholder trading losses centered (cf. the next-to-last paragraph of Section 1.2), clean and CA desks value by $\mathbb{Q}^\star$ expectation their shareholder sensitive cash flows. These include, in case of default of the bank, the transfer of property from the CA desks to the clean desks of the collateral amount MTM on the clean margin account, as well as (cf. Assumptions 1.5.ii and iii) the transfer from shareholders to bondholders of the residual value $\mathrm{RC} = \mathrm{CA}$ on the reserve capital account. Accordingly:

Definition 1.11. *We let*

$$\mathrm{MtM} = \mathbb{E}^\star(\mathcal{P}^\circ + (1 - J)\,\mathrm{MtM}) \tag{1.15}$$

and

$$\mathrm{CA} = \mathrm{CVA} + \mathrm{FVA}, \tag{1.16}$$

where

$$\mathrm{CVA} = \mathbb{E}^\star(\mathcal{C}^\circ + (1 - J)\mathrm{CVA}) \tag{1.17}$$
$$\mathrm{FVA} = \mathbb{E}^\star(\mathcal{F}^\circ + (1 - J)\mathrm{FVA}), \tag{1.18}$$

hence $\mathrm{CA} = \mathbb{E}^\star(\mathcal{C}^\circ + \mathcal{F}^\circ + (1 - J)\,\mathrm{CA})$. *We also define the contra-liabilities value*

$$\mathrm{CL} = \mathrm{DVA} + \mathrm{FDA} \tag{1.19}$$

where

$$\mathrm{DVA} = \mathbb{E}^\star(\mathcal{C}^\bullet + (1 - J)\mathrm{CVA}) \tag{1.20}$$
$$\mathrm{FDA} = \mathbb{E}^\star(\mathcal{F}^\bullet + (1 - J)\mathrm{FVA}). \tag{1.21}$$

Finally we define the firm valuation of counterparty risk,

$$\mathrm{FV} = \mathbb{E}^\star(\mathcal{C} + \mathcal{F}). \tag{1.22}$$

The definitions of MtM, CVA, and FVA are in fact fix-point equations. However, the following result shows that these equations are well-posed and yields explicit formulas for all the quantities at hand.

Proposition 1.12. *We have*

$$\mathrm{MtM} = \mathbb{E}\mathcal{P}^\circ \tag{1.23}$$
$$\mathrm{CVA} = \mathbb{E}((1 - J_1)\mathcal{P}^+) \tag{1.24}$$
$$\mathrm{FVA} = \gamma\,(\mathrm{MtM} - \mathrm{CA})^+ = \frac{\gamma}{1+\gamma}\,(\mathrm{MtM} - \mathrm{CVA})^+ \tag{1.25}$$

and

$$\mathbb{E}^\star L^\circ = \mathbb{E}L = 0 \tag{1.26}$$
$$\mathrm{FDA} = \mathrm{FVA} \tag{1.27}$$
$$\mathrm{FV} = \mathbb{E}^\star\mathcal{C} = \mathrm{CVA} - \mathrm{DVA} = \mathrm{CA} - \mathrm{CL}. \tag{1.28}$$

Proof. The first identities in each line of (1.23) follow from Definition 1.11 by Lemma 1.7 and definition of the involved cash flows in Lemma 1.9. Given (1.16), the formula $\mathrm{FVA} = \gamma\,(\mathrm{MtM} - \mathrm{CA})^+$ in (1.23) is in fact a semi-linear equation

$$\mathrm{FVA} = \gamma\,(\mathrm{MtM} - \mathrm{CVA} - \mathrm{FVA})^+. \tag{1.29}$$

But, as γ (a probability) is nonnegative, this equation has the unique solution given by the right-hand side in the third line of (1.23).

Regarding (1.26), we have

$$\mathbb{E}^\star L^\circ = (1-\gamma)\mathbb{E}((1 - J_1)\,\mathcal{P}^+ + \gamma\,(\mathrm{MtM} - \mathrm{CA})^+ - \mathrm{CA}) = 0,$$

by application of (1.4), the first line in (1.11), (1.23), and (1.16). Hence, using (1.4) again,

$$\mathbb{E}L = (1-\gamma)^{-1}\mathbb{E}^\star L^\circ = 0.$$

This is the first line in (1.26), which implies the following ones by definition of the involved quantities and from the assumption that $\mathbb{E}^\star\mathcal{F} = 0$. □

Note that $\mathrm{MtM} = \mathbb{E}\mathcal{P}^\circ$ also coincides with $\mathbb{E}\mathcal{P}$ (cf. (1.23) and the parenthesis following (1.4)). In practice $\mathcal{P}^\circ$ has less terms than $\mathcal{P}$ (that also includes cash flows from bank default onward), which is why we favor the formulation $\mathbb{E}\mathcal{P}^\circ$ in (1.23). The alternative formulation $\mathbb{E}\mathcal{P}$ may seem more in line with the intuition of MtM as value deprived from any credit/funding considerations. However, as the measure underlying $\mathbb{E}$ is the survival one (see before Lemma 1.7), this intuition is in fact simplistic and only strictly correct in the case without wrong way risk between credit and market (cf. the parenthesis preceding Lemma 1.7).

1.3.3 Capital Valuation Adjustment

Economic capital (EC) is the level of capital at risk that a regulator would like to see on an economic, structural basis. Risk calculations are typically performed by banks "on a going concern", i.e. assuming that the bank itself does not default. Accordingly:

Definition 1.13. *The economic capital (*EC*) of the bank is given by the 97.5% expected shortfall*[1.3] *of the bank trading loss L under $\mathbb{Q}$, which we denote by*[1.4] $\mathbb{ES}(L^\circ)$.

The risk margin (sized by the to-be-defined KVA in our setup) is also loss-absorbing, i.e. part of capital at risk, and the KVA is originally sourced from the client (see Assumption 1.5.iii). Hence, *shareholder* capital at risk only consists of the *difference* between the (total) capital at risk and the KVA. Accordingly (and also accounting, regarding (1.31), for the last part in Assumption 1.5.iii):

Definition 1.14. *The capital at the risk (CR) of the bank is given by* $\max(\mathrm{EC}, \mathrm{KVA})$ *and the ensuing shareholder capital at risk (SCR) by*

$$\mathrm{SCR} = \max(\mathrm{EC}, \mathrm{KVA}) - \mathrm{KVA} = (\mathrm{EC} - \mathrm{KVA})^+, \tag{1.30}$$

where, given some hurdle rate (target return-on-equity) h,

$$\mathrm{KVA} = \mathbb{E}^\star(h\,\mathrm{SCR}^\circ + (1-J)\,\mathrm{KVA}). \tag{1.31}$$

Remark 1.15. In view of (1.31) and of the last balance condition in (1.3), we have

$$\mathrm{RM}^{sh} = \mathbb{E}^\star(h\,\mathrm{SCR}^\circ)^{\mathrm{RM}^{bh}} = \mathbb{E}^\star((1-J)\,\mathrm{KVA}). \tag{1.32}$$

We refer the reader to the last bullet point in [Crépey, 2022](Definition A.1) for the analogous split of RM between shareholder and bondholder wealth in a dynamic, continuous-time setup.

Proposition 1.16. *We have*

$$\mathrm{KVA} = h\,\mathrm{SCR} = \frac{h}{1+h}\mathrm{EC} = \frac{h}{1+h}\,\mathbb{ES}(L^\circ). \tag{1.33}$$

Proof. The first identity follows from Lemma 1.7. The resulting KVA semi-linear equation (in view of (1.30)) is solved similarly to the FVA equation (1.29). □

The KVA formula (1.33) (as well as its continuous-time analog (1.71)) can be used either in the direct mode, for computing the KVA corresponding to a given h, or in the reverse-engineering mode, for defining the "implied hurdle rate" associated with the actual level on the risk margin account of the bank. Cost of capital proxies have always been used to estimate return-on-equity. The KVA is a refinement, fine-tuned for derivative portfolios, but the base return-on-equity concept itself is far older than even the CVA. In particular, the KVA is very useful in the context of collateral and capital optimization.

KVA Risk Premium and Indifference Pricing Interpretation The CA component of the FTP corresponds to the expected costs for the shareholders of concluding the deal. This CA component makes the shareholder trading loss L° centered (cf. the first line in (1.26)). On top of expected shareholder costs, the bank charges to the clients a risk margin (RM). Assume the bank shareholders endowed with a utility function U on $\mathbb{R}$ such that $U(0) = 0$. In a shareholder indifference pricing framework, the risk margin arises as per the following equation:

$$\mathbb{E}^\star U\left(J\left(\mathrm{RM} - L\right)\right) = \mathbb{E}^\star U(0) = 0 \tag{1.34}$$

(the expected utility of the bank shareholders without the deal), where

$$\mathbb{E}^\star U\left(J\left(\mathrm{RM} - L\right)\right) = \mathbb{E}^\star\left(JU\left(\mathrm{RM} - L\right)\right) = (1-\gamma)\mathbb{E}U\left(\mathrm{RM} - L\right),$$

1.3. See e.g. [Föllmer and Schied, 2016](Section 4.4).

1.4. Note that, by definition of $\mathbb{Q}$, this quantity does not depend on $L^\bullet$.

by (1.4). Hence

$$\mathbb{E}U\,(\mathrm{RM}-L)=0. \tag{1.35}$$

The corresponding RM is interpreted as the minimal admissible risk margin for the deal to occur, seen from bank shareholders' perspective.

Taking for concreteness $U\,(-\ell)=\frac{1-e^{\rho\ell}}{\rho}$, for some risk aversion parameter ρ, (1.35) yields $\mathrm{RM}=\rho^{-1}\ln\mathbb{E}e^{\rho L}=\rho^{-1}\ln\mathbb{E}e^{\rho L^\circ}$, by the observation following (1.4). In the limiting case where the shareholder risk aversion parameter $\rho\to 0$ and $\mathbb{E}U\,(-L)\to-\mathbb{E}(L)=0$ (by the first line in (1.26)), then $\mathrm{RM}\to 0$.

In view of (1.3) and (1.33), the corresponding implied KVA and hurdle rate h are such that

$$\mathrm{KVA}=\rho^{-1}\ln\mathbb{E}e^{\rho L^\circ}\,h\,1+h=\frac{\rho^{-1}\ln\mathbb{E}e^{\rho L^\circ}}{\mathbb{E}\,\mathbb{S}(L^\circ)}.$$

Hence, "for h and ρ small",

$$h\approx\frac{\mathbb{V}\mathrm{ar}(L^\circ)}{2\,\mathbb{E}\,\mathbb{S}(L^\circ)}\;\rho \tag{1.36}$$

(as $\mathbb{E}(L^\circ)=0$), where $\mathbb{V}\mathrm{ar}$ is the $\mathbb{Q}$ variance operator. The hurdle rate h in our KVA setup plays the role of a risk aversion parameter, like ρ in the exponential utility framework.

An indifference price has a competitive interpretation. Assume that the bank is competing for the client with other banks. Then, in the limit of a continuum of competing banks with a continuum of indifference prices, whenever a bank makes a deal, this can only be at its indifference price. Our stylized indifference pricing model of a KVA defined by a constant hurdle rate h exogenizes (by comparison with the endogenous hurdle rate h in (1.35)) the impact on pricing of the competition between banks. It does so in a way that generalizes smoothly to a dynamic setup (see Section 1.4), as required to deal with a real derivative banking portfolio. It then provides a refined notion of return-on-equity for derivative portfolios, where a full-fledged optimization approach would be impractical.

1.3.4 Collateral With Clients and Fungibility of Capital at Risk as a Funding Source

In case of variation margin (VM) that would be exchanged between the bank and its client, and of initial margin that would be received (RIM) and posted (PIM) by the bank, at the level of, say, some $\mathbb{Q}$ value-at-risk of $\pm(\mathcal{P}-\mathrm{VM})$, then

- $\mathcal{P}$ needs be replaced by $(\mathcal{P}-\mathrm{VM}-\mathrm{RIM})$ everywhere in the above, whence an accordingly modified (in principle: diminished) CVA,
- an additional initial margin related cash flow in $\mathcal{F}^\circ$ given as $J\gamma\,\mathrm{PIM}$, triggering an additional adjustment MVA in CA, where

$$\mathrm{MVA}=\mathbb{E}^\star(J\gamma\,\mathrm{PIM}+(1-J\,)\mathrm{MVA})=\gamma\,\mathrm{PIM}; \tag{1.37}$$

- additional initial margin related cash flows in $\mathcal{F}^\bullet$ given as $(1-J)\,(\mathrm{PIM}-\gamma\mathrm{PIM})$ and $(1-J)\mathrm{MVA}$, triggering an additional adjustment $\mathrm{MDA}=\mathrm{MVA}$ in CL;
- the second FVA formula in (1.23) modified into

$$\mathrm{FVA}=\frac{\gamma}{1+\gamma}(\mathrm{MtM}-\mathrm{VM}-\mathrm{CVA}-\mathrm{MVA})^+.$$

Accounting further for the additional free funding source provided by capital at risk (cf. Remark 1.8), then, in view of the specification given in the first sentence of Definition 1.14 for the latter, one needs replace $(\mathrm{MtM}-\mathrm{CA})^{\pm}$ by $(\mathrm{MtM}-\mathrm{VM}-\mathrm{CA}-\max(\mathrm{EC},\mathrm{KVA}))^{\pm}$ everywhere before. This results in the same CVA and MVA as in the bullet points above, but in the following *system* for the random variable L° and the FVA and the KVA numbers (cf. the corresponding lines in (1.11), (1.23), (1.33), and recall (1.16)):

$$L^\circ = J(1-J_1)\mathcal{P}^+ + J\gamma(\mathrm{MtM}-\mathrm{VM}-\mathrm{CA}-\max(\mathrm{EC},\mathrm{KVA}))^+ + J\gamma\mathrm{PIM} - J\mathrm{CA} \tag{1.38}$$

$$\mathrm{FVA} = \gamma(\mathrm{MtM}-\mathrm{VM}-\mathrm{CA}-\max(\mathrm{EC},\mathrm{KVA}))^+ \tag{1.39}$$

$$\mathrm{KVA} = \frac{h}{1+h}\mathbb{ES}(L^\circ). \tag{1.40}$$

This system entails a coupled dependence between, on the one hand, the FVA and KVA numbers and, on the other hand, the shareholder loss process L°. However, once CVA, PIM, RIM, and MVA computed as in the above, the system (1.38) can be addressed numerically by Picard iteration, starting from, say, $L^{(0)} = \mathrm{KVA}^{(0)} = 0$ and $\mathrm{FVA}^{(0)} = \frac{\gamma}{1+\gamma}(\mathrm{MtM}-\mathrm{VM}-\mathrm{CVA}-\mathrm{MVA})^+$ (cf. the last line in (1.23)), and then iterating in (1.38) until numerical convergence.

Remark 1.17. The rationale for funding FVA but not MVA from $\mathrm{CA}+\max(\mathrm{EC},\mathrm{KVA})$ is set out before Equation (15) in [Albanese et al., 2017].

1.3.5 Funds Transfer Price

The funds transfer price (all-inclusive XVA rebate to MtM) aligning the deal to shareholder interest (in the sense of a given hurdle rate h, cf. the next-to-last paragraph of Section 1.2) is

$$\mathrm{FTP} = \underbrace{\mathrm{CVA}+\mathrm{FVA}}_{\text{Expected shareholder costs CA}} + \underbrace{\mathrm{KVA}}_{\text{Shareholder risk premium}} \tag{1.41}$$

$$= \underbrace{\mathrm{CVA}-\mathrm{DVA}}_{\text{Firm valuation FV}} + \underbrace{\mathrm{DVA}+\mathrm{FDA}}_{\text{Wealth transfer CL}} + \underbrace{\mathrm{KVA}}_{\text{Shareholder Risk premium}}, \tag{1.42}$$

where all terms are explicitly given in Propositions 1.12 and 1.16 (or the corresponding variants of Section 1.3.4 in the refined setup considered there).

Wealth Transfer Analysis The above results implicitly assumed that the bank cannot hedge jump-to-default cash flows (cf. Section 1.1.1). To understand this, let us temporarily suppose, for the sake of the argument, that the bank would be able to hedge its own jump-to-default through a further deal, whereby the bank would deliver a payment $L^\bullet$ at time 1 in exchange of a fee fairly valued as

$$\mathrm{CL} = \mathbb{E}^\star L^\bullet = \mathrm{DVA}+\mathrm{FDA}, \tag{1.43}$$

deposited in the reserve capital account of the bank at time 0.

We include this hedge and assume that the client would now contribute at the level of $\mathrm{FV}=\mathrm{CA}-\mathrm{CL}$ (cf. (1.26)), instead of CA before, to the reserve capital account of the bank at time 0. Then the amount that needs be borrowed by the bank for implementing its strategy is still $\gamma(\mathrm{MtM}-\mathrm{CA})^+$ as before (back to the baseline funding setup of Remark 1.8). But the trading loss of the bank becomes, instead of L before,

$$\mathcal{C}+\mathcal{F}-\mathrm{FV}+(L^\bullet-\mathrm{CL}) = \mathcal{C}+\mathcal{F}-\mathrm{CA}+L^\bullet = L+L^\bullet = L^\circ, \tag{1.44}$$

where the last line in (1.26) and the last identity in (1.5) were used in the first and second equality. By comparison with the situation from previous sections without own-default hedge by the bank:

- the shareholders are still indifferent to the deal in expected counterparty default and funding expenses terms,

- the recovery of the bondholders becomes zero,
- the client is better off by the amount $\mathrm{CA} - \mathrm{FV} = \mathrm{CL}$.

The CL originating cash flow $L^\bullet$ has been hedged and monetized by the shareholders, who have passed the corresponding benefit to the client.

Under a cost-of-capital pricing approach, the bank would still charge to its client a KVA add-on $\frac{h}{1+h}\mathbb{E}\mathbb{S}(L^\circ)$, as risk compensation for the nonvanishing shareholder trading loss L° still triggered by the deal. If, however, the bank could also hedge the (zero-valued, by the first line in (1.26)) loss L°, hence the totality of $L = L^\circ - L^\bullet$ (instead of $L^\bullet$ only in the above), then the trading loss and the KVA would vanish. As a result, the all-inclusive XVA add-on (rebate from MtM valuation) would boil down to

$$\mathrm{FV} = \mathrm{CVA} - \mathrm{DVA}$$

(cf. (1.1)), the value of counterparty risk and funding to the bank as a whole.

Connection With the Modigliani-Miller Theory

The Modigliani-Miller invariance result, with Modigliani and Miller, 1958 as a seminal reference, consists in various facets of a broad statement that the funding and capital structure policies of a firm are irrelevant to the profitability of its investment decisions. Modigliani-Miller (MM) irrelevance, as we put it for brevity hereafter, was initially seen as a pure arbitrage result. However, it was later understood that there may be market incompleteness issues with it. So quoting [Duffie and Sharer, 1986](page 9), "generically, shareholders find the span of incomplete markets a binding constraint [...] shareholders are not indifferent to the financial policy of the firm if it can change the span of markets (which is typically the case in incomplete markets)"; or [Gottardi, 1995](page 197): "When there are derivative securities and markets are incomplete the financial decisions of the firm have generally real effects".

A situation where shareholders may "find the span of incomplete markets a binding constraint" is when market completion is legally forbidden. This corresponds to the XVA case, which is also at the crossing between market incompleteness and the presence of derivatives pointed out above as the MM non irrelevance case in Gottardi, 1995. Specifically, the contra-assets and contra-liabilities that emerge endogenously from the impact of counterparty risk on the derivative portfolio of a bank cannot be "undone" by shareholders, because jump-to-default risk cannot be replicated by a bank.

As a consequence, MM irrelevance is expected to break down in the XVA setup. In fact, as visible on the trade incremental FTP (counterparty risk pricing) formula (1.41) (cf. also (1.50) and Proposition 1.25 in a dynamic and trade incremental setup below), cost of funding and cost of capital are material to banks and need be reflected in entry prices for ensuring shareholder indifference to the trades, i.e. preserving their hurdle rate throughout trades.

1.4 XVA Analysis in a Dynamic Setup

We now consider a dynamic, continuous-time setup, with model filtration $\mathbb{G}$ and a (positive) bank default time τ endowed with an intensity γ. The bank survival probability measure associated with the measure $\mathbb{Q}^\star$ is then the probability measure $\mathbb{Q}$ with $(\mathbb{G}, \mathbb{Q}^\star)$ density process $Je^{\int_0^\cdot \gamma_s ds}$ (assumed integrable), where $J = \mathbb{1}_{[0,\tau)}$ is the bank survival indicator process (cf. [Schönbucher, 2004] and [Collin-Dufresne et al., 2004]). In particular, writing $Y^\circ = JY + (1-J)\,Y_{\tau-}$, for any left-limited process Y, we have by application of the results of Crépey and Song, 2017 (cf. the condition (A) there):

Lemma 1.18. *For every $\mathbb{Q}$ (resp. sub-, resp. resp. super-) martingale Y, the process Y° is a $\mathbb{Q}^\star$ (resp. sub-, resp. resp. super-) martingale.*

Remark 1.19. In the dynamic setup, the survival measure formulation is a light presentation, sufficient for the purpose of the present paper (skipping the related integrability issues), of an underlying reduction of filtration setup, which is detailed in the above-mentioned reference (regarding Lemma 1.18, cf. also [Collin-Dufresne et al., 2004](Lemma 1)).

1.4.1 Case of a Run-Off Portfolio

First, we consider the case of a portfolio held on a run-off basis (cf. Section 1.2.1). We denote by T the final maturity of the portfolio and we assume that all prices and XVAs vanish at time T if $T<\tau$. Then the results of Crépey, 2022 show that all the qualitative insights provided by the one-period XVA analysis of Section 1.3 are still valid. The trading loss of the bank is now given by the process

$$L=\mathcal{C}+\mathcal{F}+\mathrm{CA}-\mathrm{CA}_0 \tag{1.45}$$

and the bank *shareholder* trading loss by the $\mathbb{Q}$ (hence $\mathbb{Q}^\star$, by Lemma 1.18) martingale

$$L^\circ=\mathcal{C}^\circ+\mathcal{F}^\circ+\mathrm{CA}^\circ-\mathrm{CA}_0. \tag{1.46}$$

In (1.45)-(1.46), we have $\mathrm{CA}=\mathrm{CVA}+\mathrm{FVA}$ as in (1.16); the processes $\mathcal{C}$, $\mathcal{F}$, CVA, and FVA are continuous-time processes analogs, detailed in the case of bilateral trade portfolios in Section 1.6.1-1.6.2, of the eponymous quantities in Section 1.3 (which were constants or random variables there).

Proposition 1.20. *The core equity tier 1 capital of the bank is given by*

$$\mathrm{CET1}=\mathrm{CET1}_0-L. \tag{1.47}$$

Shareholder equity is given by

$$\mathrm{SHC}=\mathrm{SHC}_0-(L+\mathrm{KVA}-\mathrm{KVA}_0). \tag{1.48}$$

Proof. In the continuous-time setup, Assumption 1.5.i is written as

$$\mathrm{RC}+\mathrm{RM}+\mathrm{SCR}+\mathrm{UC}-\mathrm{CM}-(\mathrm{RC}+\mathrm{RM}+\mathrm{SCR}+\mathrm{UC}-\mathrm{CM})_0=\mathcal{P}-(\mathcal{C}+\mathcal{F}+\mathcal{H}).$$

Given the definition of CET1 in (1.2), the perfect clean hedge condition (see after Remark 1.3) written in the dynamic setup as $\mathcal{P}+\mathrm{MtM}-\mathrm{MtM}_0-\mathcal{H}=0$, and the balance conditions (1.3), this is equivalent to

$$\mathrm{CA}+\mathrm{CET1}-(\mathrm{CA}+\mathrm{CET1})_0=-(\mathcal{C}+\mathcal{F}).$$

In view of (1.45), we obtain (1.47).

As $\mathrm{SHC}=\mathrm{CET1}-\mathrm{RM}$ (cf. (1.2)), we have by (1.47):

$$\mathrm{SHC}=\mathrm{CET1}_0-L-\mathrm{RM}=\mathrm{CET1}_0-\mathrm{RM}_0-(L+\mathrm{RM}-\mathrm{RM}_0),$$

which, by the third balance condition in (1.3), yields (1.48). □

Moreover, by Lemma 1.18, the continuous-time process KVA that stems from (1.69)-(1.70) is a $\mathbb{Q}^\star$ supermartingale with terminal condition $\mathrm{KVA}_T=0$ on $\{T<\tau\}$ and drift coefficient $h\,\mathrm{SCR}$, where SCR is given as in (1.30), but for EC there dynamically defined as the time-t conditional, 97.5% expected shortfall of $(L^\circ_{t+1}-L^\circ_t)$ under $\mathbb{Q}$, killed at τ.

Remark 1.21. It is only before τ that the right-hand-sides in the definitions (1.2) really deserve the respective interpretations of shareholder equity of the bank and core equity tier 1 capital. Hence, it is only the parts of (1.47) and (1.48) stopped before τ, i.e.

$$\mathrm{CET1}^\circ=\mathrm{CET1}_0-L^\circ,\mathrm{SHC}^\circ=\mathrm{SHC}_0-(L^\circ+\mathrm{KVA}^\circ-\mathrm{KVA}_0), \tag{1.49}$$

which are interesting financially.

1.4.2 Trade Incremental Cost-of-Capital XVA Strategy

In [Crépey, 2022] and in Section 1.4.1 above, the derivative portfolio of the bank is assumed held on a run-off basis. By contrast, real-life derivative portfolios are incremental.

Assume a new deal shows up at time $\theta \in (0, \tau)$. We denote by $\Delta\cdot$, for any portfolio related process, the difference between the time θ values of this process for the run-off versions of the portfolio with and without the new deal.

Definition 1.22. *We apply the following trade incremental pricing and accounting policy:*

- *The clean desks pay* ΔMtM *to the client and the CA desks add an amount* ΔMtM *on*[1.5] *the clean margin account;*
- *The CA desks charge to the client an amount* ΔCA *and add it on*[1.6] *the reserve capital account;*
- *The management of the bank charges the amount* ΔKVA *to the client and adds it on*[1.7] *the risk margin account.*

The funds transfer price of a deal is the all-inclusive XVA add-on charged by the bank to the client in the form of a rebate with respect to the mark-to-market ΔMtM of the deal. Under the above scheme, the overall price charged to the client for the deal is $\Delta\text{MtM} - \Delta\,\text{CVA} - \Delta\text{KVA}$, i.e.

$$\begin{aligned} \text{FTP} &= \Delta\,\text{CVA} + \Delta\text{KVA} = \Delta\text{CVA} + \Delta\text{FVA} + \Delta\text{KVA} && (1.50)\\ &= \Delta\text{FV} + \Delta\text{CL} + \Delta\text{KVA}, && (1.51) \end{aligned}$$

by (1.16) and the last line in (1.26) (which still hold in continuous time, see [Crépey, 2022](Equations (1) and (66))) applied to the portfolios with and without the new deal.

Remark 1.23. As opposed to the ΔXVA terms, which entail portfolio-wide computations, ΔMtM reduces to the so-called clean valuation of the new deal, by trade-additivity of MtM (as follows from [Crépey, 2022](Equations (25) and (37))).

Obviously, the legacy portfolio of the bank has a key impact on the FTP. It may very well happen that the new deal is risk-reducing with respect to the portfolio, in which case $\text{FTP} < 0$, i.e. the overall, XVA-inclusive price charged by the bank to the client would be $\Delta\text{MtM} - \text{FTP} > \Delta\text{MtM}$ (subject of course to the commercial attitude adopted by the bank under such circumstance).

In order to exclude for simplicity jumps of our L and KVA processes at θ (the ones related to the initial portfolio, but also those, starting at time θ, corresponding to the augmented portfolio), we assume a quasi-left continuous model filtration $\mathbb{G}$ and a $\mathbb{G}$ predictable stopping time θ. The first assumption excludes that martingales can jump at predictable times. It is satisfied in all practical models and, in particular, in all models with Lévy or Markov chain driven jumps. The second assumption is reasonable regarding the time at which a financial contract is concluded. Note that it was actually already assumed regarding the (fixed) time 0 at which the portfolio of the bank is supposed to have been set up in the first place.

Lemma 1.24. *Assuming the new trade at time* θ *handled by the trade incremental policy of Definition 1.22 after that the balance conditions (1.3) have been held before* θ*, then shareholder equity* SHC° *(see Remark 1.21) is a* $\mathbb{Q}^\star$ *submartingale on* $[0,\theta]\cap\mathbb{R}_+$*, with drift coefficient* $h\,\text{SCR}$ *killed at* τ*.*

1.5. i.e. remove $(-\Delta\text{MtM})$ from, if $\Delta\text{MtM} < 0$.

1.6. i.e. remove $(-\Delta\text{CA})$ from, if $\Delta\text{CA} < 0$.

1.7. i.e. removes $(-\Delta\text{KVA})$ from, if $\Delta\text{KVA} < 0$.

Proof.
In the case of a trade incremental portfolio, a priori, the second identity in (1.49) is only guaranteed to hold *before* θ. However, in view of the observation made in Remark 1.6 and because, under our (harmless) technical assumptions, there can be no dividends arising from the portfolio expanded with the new deal (i.e. jumps in the related processes L and KVA, defined on $[\theta, +\infty)$) at time θ itself, the process SHC does not jump at θ. The process L and KVA related to the legacy portfolio cannot jump at θ either. As a result, the second identity in (1.49) still holds at θ. It is therefore valid on $[0, \theta] \cap \mathbb{R}_+$. The result then follows from the respective martingale and supermartingale properties of the (original) processes L° and KVA recalled before and after Proposition 1.20. □

The above XVA strategy can be iterated between and throughout every new trade. We call this approach the ***trade incremental cost-of-capital XVA strategy***. By an iterated application of Lemma 1.24 at every new trade, we obtain the following:

Proposition 1.25. *Under a dynamic and trade incremental cost-of-capital XVA strategy, shareholder equity* SHC° *is a* $\mathbb{Q}^\star$ *submartingale on* $\mathbb{R}_+$*, with drift coefficient* $h\,\mathrm{SCR}$ *killed at* τ*.*

Thus, a trade incremental cost-of-capital XVA strategy results in a sustainable strategy for profits retention, both between and throughout deals, which was already the key principle behind Solvency II (see Section 1.1.1). Note that, without the KVA (i.e. for $h = 0$), the (risk-free discounted) shareholder equity process SHC° would only be a $\mathbb{Q}^\star$ martingale, which could only be acceptable to shareholders without risk aversion (cf. Section 1).

1.4.3 Computational Challenges

Figure 1.2 yields a picturesque representation, in the form of a corresponding XVA dependence tree, of the continuous-time XVA equations.

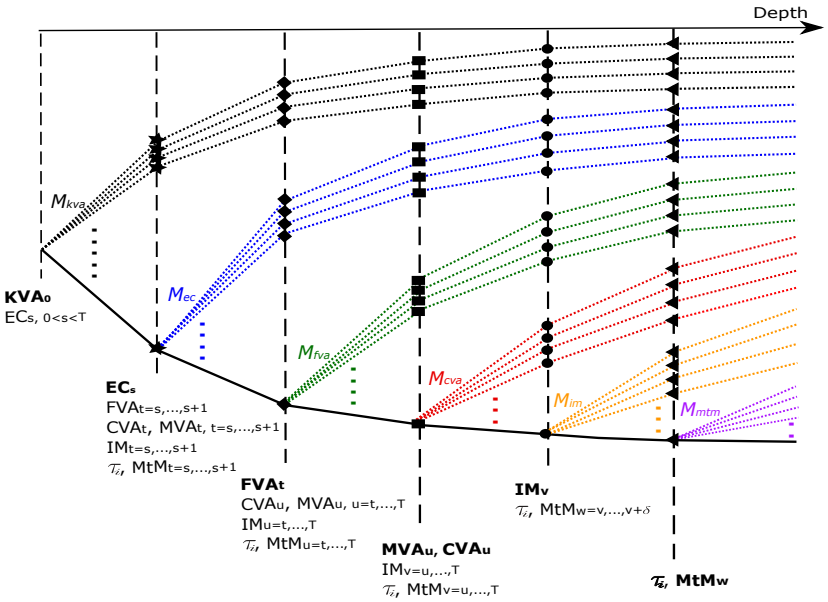

Figure 1.2. The XVA equations dependence tree (*Source*: [Abbas-Turki et al., 2018]).

For concreteness, we restrict ourselves to the case of bilateral trading in what follows, referring the reader to [Albanese et al., 2020](Section 6.2) for the more general and realistic situation of a bank also involved in centrally cleared trading. As visible from the corresponding equations in Section 1.6, the CVA of the bank can then be computed as

the sum of its CVAs restricted to each netting set (or counterparty i of the bank, with default time denoted by τ_i in Figure 1.2). The initial margins and the MVA are also most accurately calculated at each netting set level. By contrast, the FVA is defined in terms of a semilinear equation that can only be solved at the level of the overall derivative portfolio of the bank. The KVA can only be computed at the level of the overall portfolio and relies on conditional risk measures of future fluctuations of the shareholder trading loss process L°, which itself involves future fluctuations of the other XVA processes (as these are part of the bank liabilities).

Moreover, the fungibility of capital at risk with variation margin (cf. Remark 1.17) induces a coupling between, on the one hand, the "backward" FVA and KVA processes and, on the other hand, the "forward" shareholder loss process L°. As in the static case of Section 1.3.4 (cf. the last paragraph there), the ensuing forward backward system can be decoupled by Picard iteration.

These are heavy computations encompassing all the derivative contracts of the bank. Yet these computations require accuracy so that trade incremental XVA computations, which are required as XVA add-ons to derivative entry prices (cf. Section 1.4.2), are not in the numerical noise of the machinery.

As developed in [Abbas-Turki et al., 2018](Section 3.2), computational strategies for (each Picard iteration of) the XVA equations involve a mix of nested Monte Carlo (NMC) and of simulation/regression schemes, optimally implemented on GPUs. In view of Figure 1.2, a pure NMC approach would involve five nested layers of simulation (with respective numbers of paths $M_{xva} \sim \sqrt{M_{mtm}}$, see [Abbas-Turki et al., 2018](Section 3.3)). Moreover, nested Monte Carlo implies intensive repricing of the mark-to-market cube, i.e. pathwise MtM valuation for each netting set, or/and high dimensional interpolation. In this work, we use no nested Monte Carlo or conditional repricing of future MtM cubes: beyond the base MtM layer in the XVA dependence tree, each successive layer (from right to left in Figure 1.2, at each Picard iteration) will be "learned" instead.

1.4.4 Deep (Quantile) Regression XVA Framework

We denote by $\mathbb{E}_t$, $\mathbb{V}a\mathbb{R}_t$, and $\mathbb{ES}_t$ (and simply, in case $t=0$, $\mathbb{E}$, $\mathbb{V}a\mathbb{R}$, and $\mathbb{ES}$) the time-t conditional expectation, value-at-risk, and expected shortfall with respect to the bank survival measure $\mathbb{Q}$.

We compute the mark-to-market cube using CUDA routines. The pathwise XVAs are obtained by deep learning regression, i.e. extension of Longstaff and Schwartz, 2001 kind of schemes to deep neural network regression bases as also considered in [Huré et al., 2020] or [Beck et al., 2019], based on the classical quadratic (also known as mean square error, MSE) loss function. The conditional value-at-risks and expected shortfalls involved in the embedded pathwise EC and IM computations are obtained by deep quantile regression, as follows.

Given features X and labels Y (random variables), we want to compute the conditional value-at-risk and expected shortfall functions $q(\cdot)$ and $s(\cdot)$ such that $\mathbb{V}a\mathbb{R}(Y|X) = q(X)$ and $\mathbb{ES}(Y|X) = s(X)$. Recall from [Fissler et al., 2016] and [Fissler and Ziegel, 2016] that value-at-risk is *elicitable*, expected shortfall is not, but their pair is *jointly elicitable*. Specifically, we consider loss functions ρ of the form (where in our notation Y is a signed loss, whereas it is a signed gain in their paper)

$$\rho(q(\cdot), s(\cdot); X, Y) = (1-\alpha)^{-1}(f(Y) - f(q(X)))^+ + f(q(X)) + \tag{1.52}$$

$$g(s(X)) - \dot{g}(s(X))\,(s(X) - q(X) - (1-\alpha)^{-1}(Y - q(X))^+). \tag{1.53}$$

One can show (cf. also [Dimitriadis and Bayer, 2019]) that, for a suitable choice of the functions f, g including $f(z) = z$ and $g = -\ln(1+e^{-z})$ (our choice in our numerics), the

pair of the conditional value-at-risk and expected shortfall functions is the minimizer, over all measurable pair-functions $(q(\cdot), s(\cdot))$, of the error

$$\mathbb{E}\rho(q(\cdot), s(\cdot); X, Y). \tag{1.54}$$

In practice, one minimizes numerically the error (1.54), based on m independent simulated values of (X, Y), over a parametrized family of functions $(q, s)(x) \equiv (q, s)_\theta(x)$. Dimitriadis and Bayer, 2019 restrict themselves to multilinear functions. In our case we use a feedforward neural network parameterization (see e.g. [Goodfellow et al., 2016]). The minimizing pair $(q, s)_{\hat{\theta}}$ then represents the two scalar neural network approximations of the conditional value-at-risk and expected shortfall functions pair.

The left and right panels of Figure 1.3 show the respective deep neural networks for pathwise value-at-risk/expected shortfall (with error (1.54)) and pathwise XVAs (with classical quadratic norm error). Deep learning methods often show particularly good generalization and scalability performances (cf. Section 1.5.5). In the case of conditional value-at-risk and expected shortfall computations, deep learning quantile regression is also easier to implement than more naive methods, such as the resimulation and sort-based scheme of Barrera et al., 2019 for the value-at-risk and expected shorfall at each outer node of a nested Monte Carlo simulation.

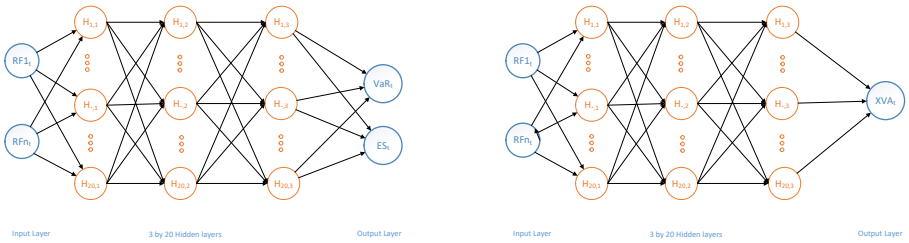

Figure 1.3. Neural networks with state variables (realizations of the risk factors at the considered pricing time) as features. *(Left)* Joint value-at-risk/expected shortfall neural network: output is joint estimate of pathwise conditional value-at-risk and expected shorfall, at a selected confidence level, of the label (inputs to initial margin or economic capital) given the features. *(Right)* XVAs neural network: output is estimate of pathwise conditional mean of the label (XVA generating cash flows) given the features.

The neural network topology and hyper-parameters used by default in our examples are detailed in Table 1.8. We use hyperbolic tangent activation functions in all cases.

	CVA	FVA	IM	MVA	Gap CVA[1.8]	EC	KVA
Hidden Layers	3	5	3	3	3	3	3
Hidden Layer Size	20	6	20	20	20	20	20
Learning Rate	0.025	0.025	0.05	0.1	0.1	0.025	0.1
Momentum	0.95	0.95	0.5	0.5	0.5	0.95	0.5
Iterations	100	50	150	100	100	100	100
Loss Function	MSE	MSE	(1.52)	MSE	(1.52)	(1.52)	MSE
Application	netting set	portf.	netting set	netting set	netting set	portf.	portf.

Table 1.2. Neural network topology and learning parameters used by default in our numerics (portf. $\equiv$ overall derivative portfolio of the bank).

Algorithm 1.1 yields our fully (time and space) discrete scheme for simulating the Picard iteration (1.74) until numerical convergence to the XVA processes. Note that, as opposed to more rudimentary, expected exposure based XVA computational approaches (see Section 1 in [Abbas-Turki et al., 2018]), this algorithm requires the simulation of the counterparty defaults.

Algorithm 1.1
Deep XVAs algorithm.

- Simulate forward m realizations (Euler paths) of the market risk factor processes and of the counterparty survival indicator processes (i.e. default times) on a refined time grid;
- For each pricing time $t = t_i$ of a pricing time grid, with coarser time step denoted by h, and for each counterparty c:
  - Learn the corresponding $\mathbb{V}a\mathbb{R}_t$ and $\mathbb{ES}_t$ terms visible in (1.75) or (under the time-discretized outer integral in) (1.77);
  - Learn the corresponding $\mathbb{E}_t$ terms visible in (1.76) through (1.78);
  - Compute the ensuing pathwise CVA and MVA as per (1.76)–(1.78);
- For $\mathrm{FVA}^{(0)}$, consider the following time discretization of (1.73) (in which λ is the risky funding spread process of the bank) with time step h:

$$\mathrm{FVA}_t^{(0)} \approx \mathbb{E}_t[\mathrm{FVA}_{t+h}^{(0)}] + h\,\bar{\lambda}_t \left(\sum_c J_t^c \left(P_t^c - \mathrm{VM}_t^c\right) - \mathrm{CVA}_t - \mathrm{MVA}_t - \mathrm{FVA}_t^{(0)} \right)^+ \tag{1.55}$$

  and, for each $t = t_i$, learn the corresponding $\mathbb{E}_t$ in (1.55), then solve the semi-linear equation for $\mathrm{FVA}_t^{(0)}$;
- For each Picard iteration k (until numerical convergence), simulate forward $L^{(k)}$ as per the first line in (1.74) (which only uses known or already learned quantities), and:
  - For economic capital $\mathrm{EC}^{(k)}$, for each $t = t_i$, learn $\mathbb{ES}_t((L^{(k)})^\circ_{t+1} - (L^{(k)})^\circ_t)$ (cf. Definition 1.27);
  - $\mathrm{KVA}^{(k)}$ and $\mathrm{FVA}^{(k)}$ then require a backward recursion solved by deep learning approximation much like the one for $\mathrm{FVA}^{(0)}$ above.

1.5 Swap Portfolio Case Study

We consider an interest rate swap portfolio case study with counterparties in different economies, first involving 10 one-factor Hull White interest-rates, 9 Black-Scholes exchange rates, and 11 Cox-Ingersoll-Ross default intensity processes. The default times of the counterparties and the bank itself are jointly modeled by a "common shock" or dynamic Marshall-Olkin copula model as per [Crépey et al., 2014](Chapt. 8–10) and [Crépey and Song, 2016] (see also Elouerkhaoui (2007, 2017)). This whole setup results in about 40 risk

factors used as deep learning features (including the counterparty default indicators).

In this model we consider a bank portfolio of 10K randomly generated swap trades, with

- trade currency and counterparty both uniform on $[1, 2, 3 \ldots, 10]$,
- notional uniform on $[10\,K, 20\,K, \ldots, 100\,K]$,
- collateralization (cf. Section 1.6.4): either "no CSA counterparty" without initial margin (IM) nor variation margin (VM), or "CSA counterparty" with VM = MtM and posted initial margin (PIM) pledged at 99% gap risk value-at-risk, received initial margin (RIM) covering 75% gap risk and leaving excess as residual gap CVA,
- for economic capital, 97.5% expected shortfall of 1-year ahead trading loss of the bank shareholders.

By default we use Monte Carlo simulation with 50K paths of 16 coarse (pricing) and 32 fine (risk factors) time steps per year.

1.5.1 Validation Results

The validation of our deep learning methodology is done in the setup of a portfolio of swaps issued at par, with final maturity $T = 10$ years, without initial margin (IM) nor variation margin (VM).

We first focus on the CVA, as the latter is amenable to validation by a standard nested Monte Carlo ("NMC") methodology. Figures 1.4, 1.5 and 1.6 show that the learned CVA is consistent with that obtained from a nested Monte Carlo simulation. Regarding Figure 1.6 (and also later below), note the equivalence of optimising the mean quadratic error

- between the neural network-learned estimator $h(X)$ and the labels Y ("MSE"), $\mathbb{E}[(h(X) - Y)^2]$, and
- between the neural network-learned estimator and the conditional expectation $\mathbb{E}[Y|X]$ (in our case estimated by NMC), $\mathbb{E}[(h(X) - \mathbb{E}[Y|X])^2]$.

The equivalence stems from the following identities, which hold for any random variables X, Y and hypothesis function h such that Y and $h(X)$ are square integrable:

$$\begin{aligned} \mathbb{E}[(h(X) - Y)^2] &= \mathbb{E}[(h(X) - \mathbb{E}[Y|X])^2] + \mathbb{E}[(\mathbb{E}[Y|X] - Y)^2] \\ &\quad + 2\,\mathbb{E}\left[(h(X) - \mathbb{E}[Y|X])\,(\mathbb{E}[Y|X] - Y)\right] \\ &= \mathbb{E}[(h(X) - \mathbb{E}[Y|X])^2] + \mathbb{E}[\mathbb{V}\mathrm{ar}(Y|X)] \end{aligned} \tag{1.56}$$

(as the second line vanishes), where $\mathbb{E}[\mathbb{V}\mathrm{ar}(Y|X)]$ does not depend on h.

The CVA error profile on Figure 1.6 reveals slightly more difficulty in learning the earlier CVAs. This is because of a higher variance of the corresponding cash flows (integrated over longer time frames) in conjunction with a lower variance of the features (risk factors diffused over shorter time horizons).

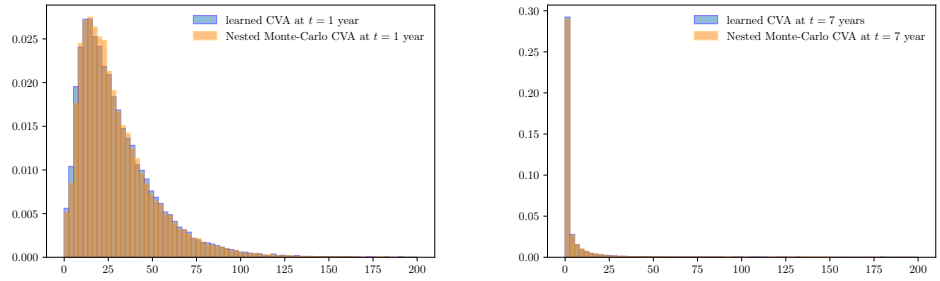

Figure 1.4. Random variables CVA_1^c and CVA_7^c (in the case of a no CSA netting set c, respectively observed after 1 and 7 years) obtained by learning (blue histogram) versus nested Monte Carlo (orange histogram). All histograms are based on out-of-sample paths.

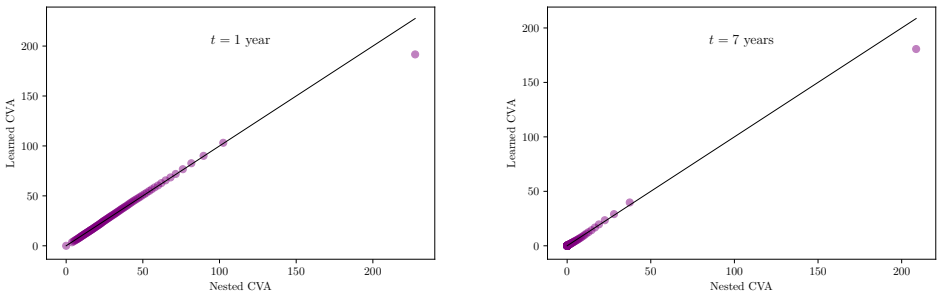

Figure 1.5. QQ-plot of learned versus nested Monte Carlo CVA for the random variables CVA_1^c (*left*) and CVA_7^c (*right*). Paths are out-of-sample.

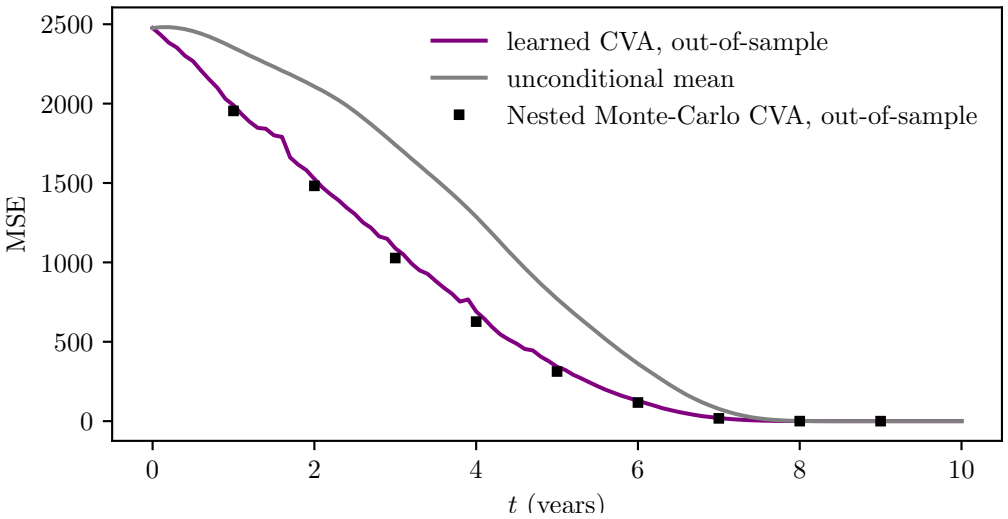

Figure 1.6. Empirical quadratic loss of each CVA estimator at all coarse time-steps. The lower, the closer to the true conditional expectation (cf. (1.56)). Since the nested Monte Carlo method is computationally expensive, it was carried out only once every 10 coarse time-steps.

Table 1.3 shows the computational cost and accuracy of the nested Monte Carlo method

for different number of inner paths, using 32768 outer paths. The convergence is already achieved for approximately 128 inner paths, in line with the NMC square root rule that is recalled in an XVA setup in [Abbas-Turki et al., 2018](Section 3.3). Figure 1.7 and Table 1.4 show that a good accuracy can be achieved through learning at a lower computational cost than through nested Monte Carlo, while also enjoying the advantages of the approach being parametric. Indeed, once the CVA is learned, one would pay only the cost of inference later on, which is generally negligible compared to training time. By contrast, a nested Monte Carlo approach would require to relaunch the nested simulations every time the CVA estimator is needed on new paths. Early stopping could be used to help reduce training time further while improving regularization.

# of inner paths	MSE (vs labels)	Computational time (seconds)
2	0.523	37.562
4	0.427	37.815
8	0.393	37.819
16	0.370	38.988
32	0.360	40.707
64	0.353	57.875
128	0.348	157.536
256	0.349	301.406
512	0.348	584.475
1024	0.348	1213.756

Table 1.3. Accuracy and computation times for the estimation of a CVA at a given coarse time-step using the nested Monte Carlo procedure. The MSE here is the mean quadratic error between the nested Monte Carlo estimator and the labels, and hence quantifies how well it is doing as a projection.

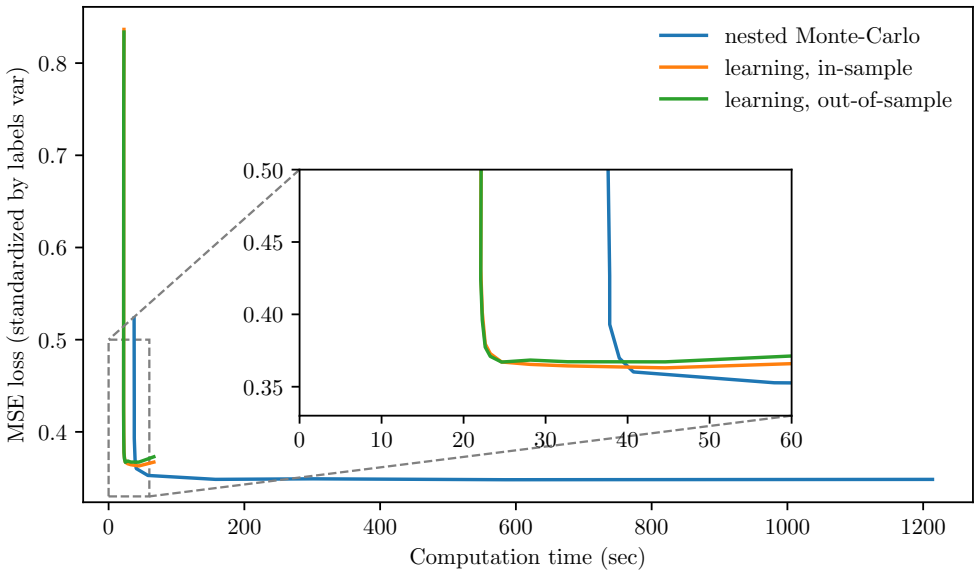

Figure 1.7. Speed versus accuracy in the case of a CVA at a given pricing time. We kept varying the number of inner paths for the nested Monte Carlo estimator and the number of epochs for the learning approach and recorded the computation time and the empirical quadratic loss.

# of epochs	MSE (vs NMC CVA)	MSE (vs labels)	Simulation time	Training time
1	0.977	0.979	21.992	0.880
2	0.729	0.729	21.992	0.434
4	0.423	0.425	21.992	0.524
8	0.399	0.401	21.992	0.719
16	0.371	0.369	21.992	1.088
32	0.369	0.365	21.992	1.800
64	0.370	0.363	21.992	3.243
128	0.371	0.363	21.992	6.227
256	0.370	0.361	21.992	10.883
512	0.370	0.362	21.992	20.096
1024	0.371	0.362	21.992	39.338

Table 1.4. Accuracy and computation times (in sec) for the calculation of a CVA at a given coarse time-step using the learning approach. MSE against NMC CVA is the mean quadratic error between the learned CVA and a CVA obtained using a nested Monte Carlo with 512 inner paths, while MSE against labels designates the mean quadratic error between the learned CVA and the labels that were used during training and thus quantifies how well it is doing as a projection. Both errors are respectively normalized by the variances of the nested Monte Carlo estimator and of the labels. The paths used here are out-of-sample.

More generally, in the presence of a multiple number of XVA layers (cf. Figure 1.2), a purely nested Monte Carlo approach would require multiple layers of nested simulations, which would amount to a computational time that is exponential in the number of XVA layers, while the computational complexity for the learning approach is linear.

As with mainstream interpolation (as opposed to regression in our case) learning problems, a good architecture is key to better learning and hence better approximation of our XVA metrics. As expected, increasing the model capacity reduces the in-sample error as shown in the bottom panel of Figure 1.8. Although fine-tuning in our case suggests a single layer yields the best out-of-sample performance for the CVA, a standard guess such as 3 layers can also be considered good enough as shown in the top panel. Of course such conclusions may depend on the complexity of the portfolio and the number of counterparties

and risk factors.

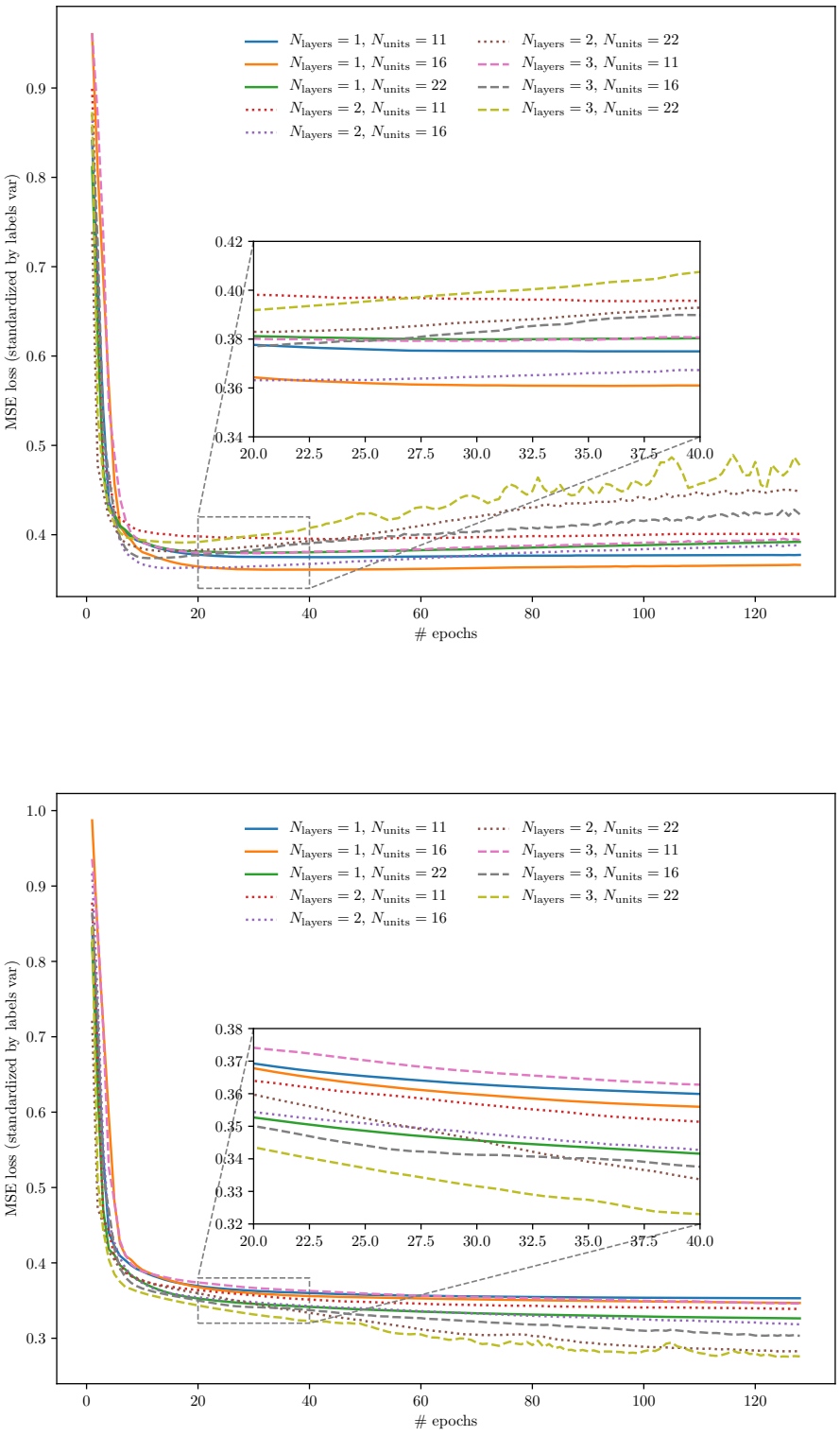

Figure 1.8. Empirical quadratic loss during CVA learning at time-step $t = 5$years, standardized by the variance of the labels. *(Bottom)* Paths are in-sample. *(Top)* Paths are out-of-sample.

Figure 1.9 shows the learned $\mathrm{FVA}^{(0)}$ profile as per (1.55). The orange FVA curve represents the mean FVA originating cash flows, which, in principle as on the picture, matches the blue mean FVA itself learned from these cash flows. The 5th and 95th percentiles FVA estimates are a bit less smooth in time then the mean profiles, as expected.

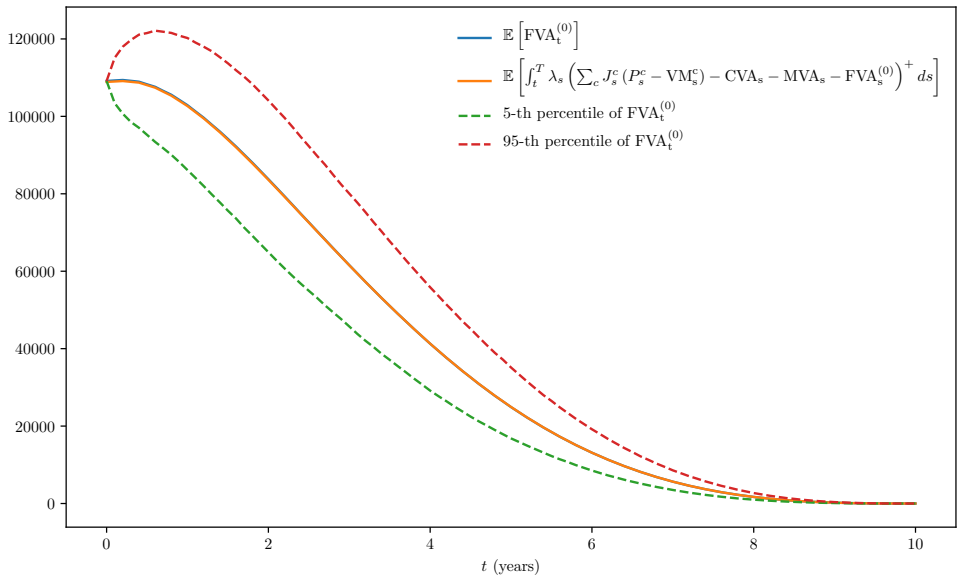

Figure 1.9. Learned $\mathrm{FVA}^{(0)}$.

Figure 1.10 *(left)* is a sanity check that the profiles of the successives iterates $L^{(k)}$ of the shareholder trading loss process L° in Algorithm 1.1 converge rapidly with k. Figure 1.10 *(right)* shows the loss process $L^{(3)}$, displayed as its mean and mean $\pm$ 2 stdev profiles. Consistent with its martingale property, the loss process $L^{(3)}$ appears numerically centered around zero. The latter holds, at least, beyond $t \sim 5$ years. For earlier times, the regression errors, accumulated backward across pricing times since the final maturity of the portfolio, induce a non negligible bias (the corresponding confidence intervals no longer contains 0). This is the reason why we use a coarser pricing time step than simulation time step in Algorithm 1.1.

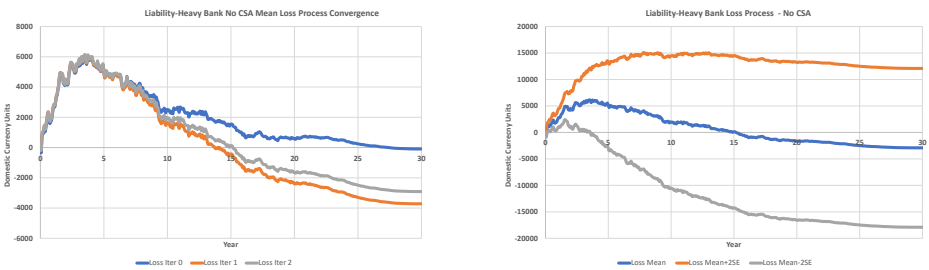

Figure 1.10. *(Left)* Profiles of the processes $L^{(k)}$, for $k = 1, 2, 3$; *(Right)* Mean $\pm$ 2 stdev profiles of the process $L^{(3)}$.

1.5.2 Portfolio-wide XVA Profiles

For the financial case study that follows, we consider

- swap rates uniformly distributed on $[0.005, 0.05]$ (hence swaps already in-the-money or out-of-the-money at time 0),
- number of six-monthly coupon resets uniform on $[5 \cdots 60]$ (final maturity of the portfolio $T = 30$ years),
- portfolio direction: either "asset heavy" bank mostly in the receivables in the future, or "liability-heavy" bank mostly in the payables in the future (respectively corresponding, with our data, to a bank 75% likely to pay fixed in the swaps, or 75% likely to receive fixed).

The figures that follow only display profiles, i.e. term structures, that is, expectations as a function of time of the corresponding processes. But all these processes are computed pathwise, based on the deep learning regression and quantile regression methodology of Section 1.4.4, allowing for all XVA inter-dependencies. Of course, XVA profiles (or pathwise XVAs if wished) are much more informative for traders than the spot XVA values (or time 0 confidence intervals) returned by most XVA systems.

Assuming 10 counterparties, Figure 1.11 shows the GPU generated profiles of

$$\mathrm{MtM} = \sum_c P^c \mathbb{1}_{[0,\tau_c^\delta)} \tag{1.57}$$

in the case of the asset-heavy portfolio and of the liability-heavy portfolio.

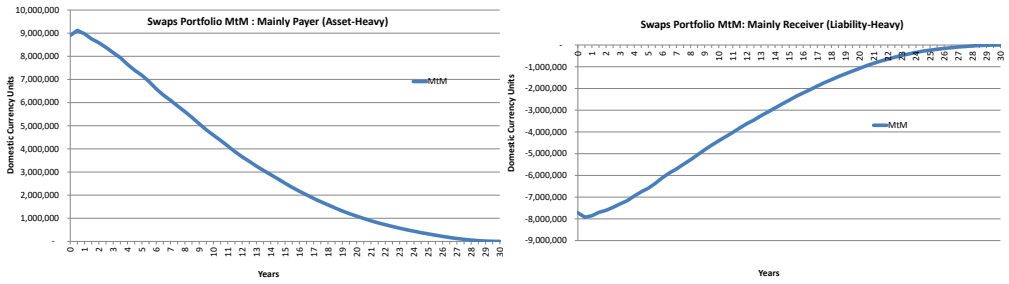

Figure 1.11. MtM profiles. *(Left)* Asset-heavy portfolio. *(Right)* Liability-heavy portfolio.

Figure 1.12 shows the porftolio-wide XVA profiles of the asset-heavy *(top)* vs. liability–heavy *(bottom)* portfolio and of the no CSA *(left)* vs. CSA portfolio *(right)*. Obviously, asset–heavy or no CSA means more CVA. The correponding curves also emphasize the transfer from counterparty credit into liquidity funding risk prompted by extensive collateralisation. Yet FVA/MVA risk is ignored in current derivatives capital regulation.

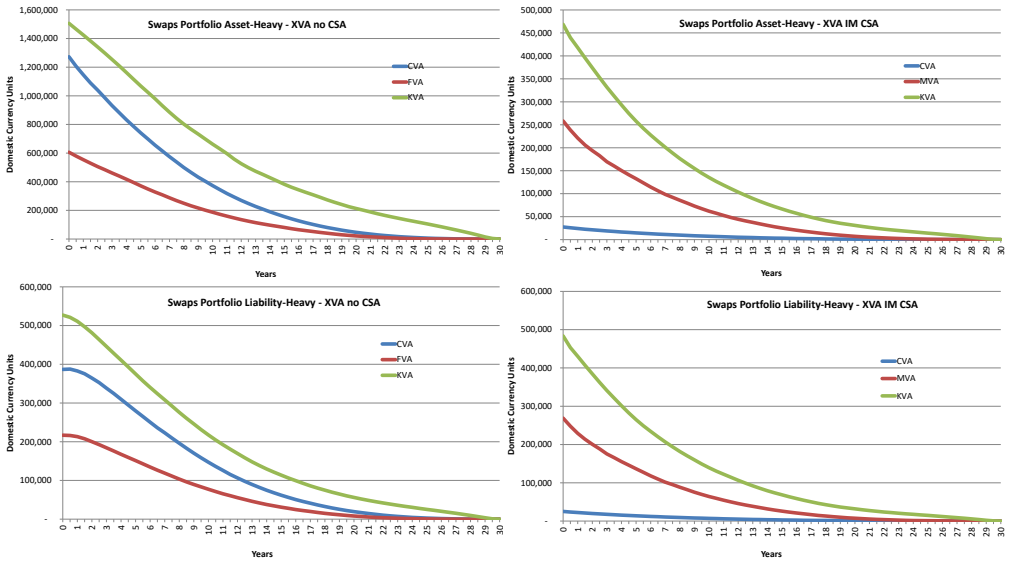

Figure 1.12. *(Top left)* Asset-heavy portfolio, no CSA. *(Top right)* Asset-heavy portfolio under CSA. *(Bottom left)* Liability–heavy portfolio, no CSA. *(Bottom right)* Liability-heavy portfolio under CSA.

Figure 1.13 shows that *(top left)* capital at risk as funding (cf. Section 1.3.4) has a material impact on the already (reserve capital as funding) reduced FVA, *(top right)* treating KVA as a risk margin (cf. (1.31)) gives a huge discounting impact, *(bottom left)* deep learning detects material initial margin convexity in the asset-heavy CSA portfolio, and *(bottom right)* deep learning detects material economic capital convexity in the asset-heavy no CSA portfolio.

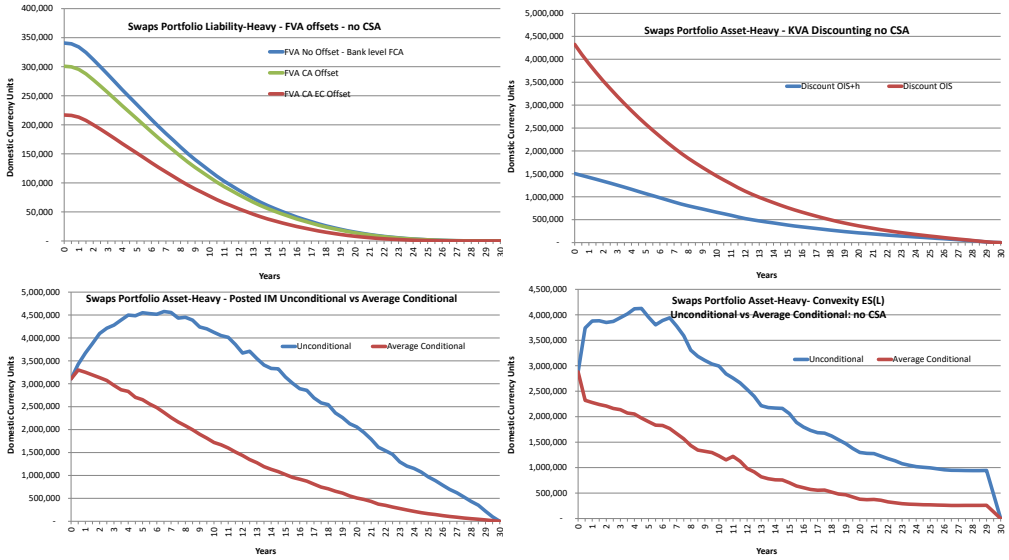

Figure 1.13. *(Top left)* FVA ignoring the off-setting impact of reserve capital and capital at risk, cf. Section 1.3.4 (blue), FVA as per (1.73) accounting for the off-setting impact of reserve capital but ignoring the one of capital at risk (green), refined FVA as per (1.65) accounting for both impacts (red). *(Top right)* KVA ignoring the off-setting impact of the risk margin, i.e. with CR instead of $(\mathrm{CR}-\mathrm{KVA})$ in (1.71) (red), refined KVA as per (1.69)–(1.70) (blue). *(Bottom left)* In the case of the asset-heavy portfolio under CSA, unconditional PIM profile, i.e. with $\mathbb{V}a\mathbb{R}_t$ replaced by $\mathbb{V}a\mathbb{R}$ in (1.75) (blue), vs. pathwise PIM profile, i.e. mean of the pathwise PIM process as per (1.75) (red). *(Bottom right)* In the asset-heavy portfolio no CSA case, unconditional economic capital profile, i.e. EC profile ignoring the words "time-t conditional" in Definition 1.27 (blue), vs. pathwise economic capital profile, i.e. mean of the pathwise EC process as per Definition 1.27 (red).

The above findings demonstrate the necessity of pathwise capital and margin calculations for accurate FVA, MVA, and KVA calculations.

1.5.3 Trade Incremental XVA Profiles

Next, we consider, on top of the previous portfolios, an incremental trade given as a par 30 year (receive fix or pay fix) swap with 100K notional. Figure 1.14 shows the trade incremental XVA profiles produced by our deep learning approach. Note that, for obtaining such smooth incremental profiles, it has been key to use common random numbers, as much as possible, between the original portfolio XVA computations and the ones regarding the portfolio expanded with the new trade.

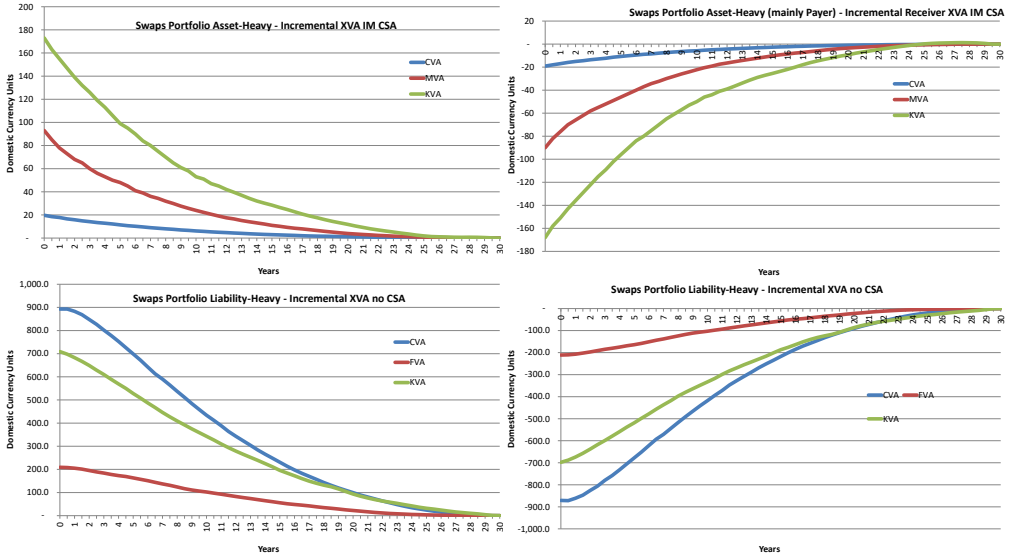

Figure 1.14. *(Top left)* Asset-heavy portfolio, no CSA. Incremental receive fix trade. *(Top right)* Liability-heavy portfolio, no CSA. Incremental pay fix trade. *(Bottom left)* Asset-heavy portfolio under CSA. Incremental Pay Fix Trade. *(Bottom right)* Liability-heavy portfolio under CSA. Incremental receive fix trade.

1.5.4 Trade and Hedge Incremental XVA Profiles

Our model assumes the market risk of trades to be fully hedged (see the paragraph following Remark 1.3 and the proofs of Lemma 1.9 and Proposition 1.20). In the previous subsection, the new swap was implicitly meant to be hedged, in terms of market risk, by the clean desks, through an accordingly modified hedging loss process $\mathcal{H}$ (see Section 1.2.1). Here we consider an alternative situation where the market risk of the new swap is back-to-back hedged via a financial, hedge counterparty. Specifically, we deal with

- 10 counterparties: 8 no CSA clients and 2 bilateral VM/IM CSA hedge counterparties,

- portfolios of 5K randomly generated swap trades as before, plus 5K corresponding hedge trades,

- an incremental trade given as a par 30 year swap with 100K notional, along with the corresponding hedge trade.

In particular, $\mathrm{MtM}_0 = 0$ (cf. (1.57)), in both portfolios excluding or including the new swap. In case a client or hedge counterparty defaults, the corresponding market hedge is assumed to be rewired through the clean desks via an accordingly modified hedging loss process $\mathcal{H}$.

The 8 no CSA counterparties are primarily asset or liability heavy. One bilateral CSA hedge counterparty is asset-heavy and one liability-heavy. Figure 1.15 provides the trade incremental XVA profiles of the bilateral hedge alternatives in combination with those for the initial counterparty trade. The main XVA impact of the hedge is then a corresponding incremental MVA term, which can contribute to make the global FTP related to the trade+hedge package more or less positive or negative, depending on the data (cf. the four panels in Figure 1.15), as can only be inferred by a refined XVA computation.

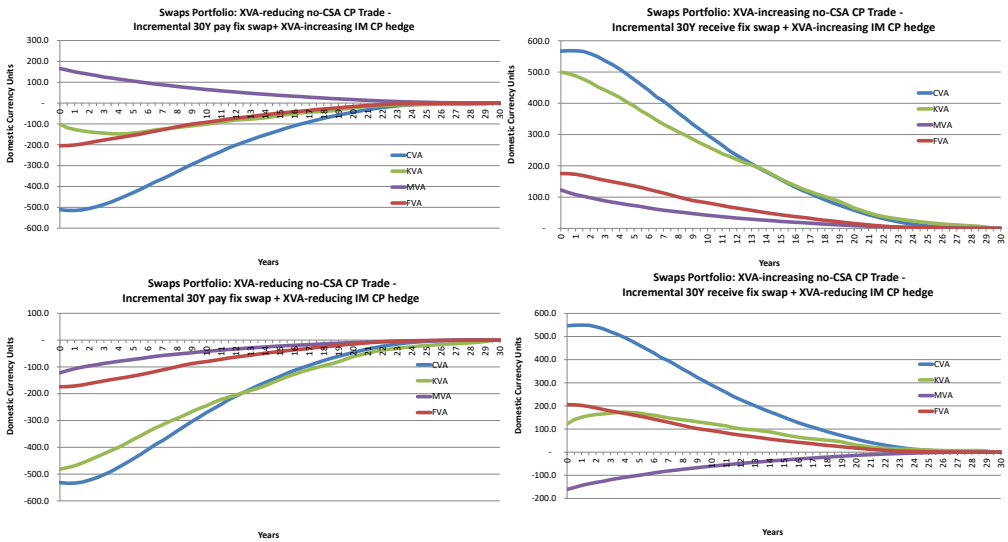

Figure 1.15. *(Top left)* XVA-reducing trade + XVA-increasing bilateral hedge *(Top right)* XVA-increasing trade + XVA-increasing bilateral hedge. *(Bottom left)* XVA-reducing trade + xva-reducing bilateral hedge *(Bottom right)* XVA-increasing trade + XVA-reducing bilateral hedge.

Remark 1.26. In the above, we do not include the XVA costs/benefits of the bilateral hedge counterparty itself. Given Remark 1.6, in different circumstances it may be possible to attribute them to client trades of the original or hedge bank. Space is lacking for a fuller discussion of economics of XVA trading in different setups. In particular, many hedge trades now face central instead of bilateral counterparties. This occurs at additional XVA costs for the client of the initial swap that can be computed the way explained in [Albanese et al., 2020].

1.5.5 Scalability

Our deep learning XVA implementation uses CNTK, the Microsoft Cognitive Toolkit. CNTK is written in core C++/CUDA (with wrappers for Python, C#, and Java). This is convenient for XVA applications, which are usually developed in C++: CNTK automatic differentiation in C++/CUDA enables C++ in-process training. This allows embedding the deep learning task within XVA processing.

Table 1.5 sets out computation times, including additional results obtained by doubling the numbers of counterparties and risk factors (to 20 counterparties and 80 risk factors).

	10 CP 40 risk factors		20 CP 80 risk factors	
	No CSA	IM CSA	No CSA	IM CSA
Initial risk factor & trade pricing simulation Cuda	352	352	426	426
Counterparty and bank level learning calculations	4,529	13,466	19,154	59,342
Total initial batch	4,881	13,818	19,580	59,768
Re-simulate 1 counterparty trade pricing Cuda	40	40	51	51
Counterparty and bank level learning calculations	2,785	2,736	7,694	6,628
Total incremental trade	2,825	2,776	7,745	6,679

Table 1.5. XVA deep learning computation timings (seconds).

All these results were based on 50K simulation paths, 32 time steps per year for risk factor simulation, and 16 time steps per year for all XVA calculations and deep learning. They were computed on a Lenovo P52 laptop with NVidia Quadro P3200 GPU @ 5.5 Teraflops peak FP32 performance, and 14 streaming multiprocessors.

The computations for 20 counterparties took more than twice as long as those for 10 counterparties. However, our deep learning calculations achieved around 80 to 90% Cuda occupancy for 10 counterparties and at times fell to half that level for 20 counterparties. Scaling to realistically high dimensions should be achievable, but acceptable trade incremental pricing performance in production would require server-grade GPU hardware, performance tuning for high GPU utilisation, and, possibly, caching computations.

1.6 Continuous-Time XVA Equations

We recall from [Crépey et al., 2020] the continuous-time XVA equations for bilateral trade portfolios when capital at risk is deemed fungible with variation margin, also adding here initial margin and MVA as in the refined static setup of Section 1.3.4.

We write $\boldsymbol{\delta}_\eta(dt) = \mathrm{d}\mathbb{1}_{\{\eta \le t\}}$ for the Dirac measure at a random time η.

1.6.1 Cash Flows

We suppose that the derivative portfolio of the bank is partitioned into bilateral netting sets of contracts which are jointly collateralized and liquidated upon bank or counterparties (whether these are clients or market hedge counterparties) default. Given a netting set c of the `bank` portfolio, we denote by:

- $\mathcal{P}^c$ and P^c, the corresponding contractually promised cash flows and clean value processes;
- τ_c, J^c, and R_c, the corresponding default times, survival indicators, and recovery rates, whereas τ, J, and R are the analogous data regarding the bank itself, with bank credit spread process $\lambda = (1-R)\,\gamma$ taken as a proxy of its risky funding spread process[1.9];
- $\tau_c^\delta = \tau_c + \delta$ and $\tau^\delta = \tau + \delta$, where δ is a positive margin period of risk, in the sense that the liquidation of the netting set c happens at time $\tau_c^\delta \wedge \tau^\delta$;

1.9. See [Albanese et al., 2020](Section 5) for the discussion of cheaper funding schemes for initial margin.

- VM^c, the variation margin (re-hypothecable collateral) exchanged between the bank and counterparty c, counted positively when received by the bank;
- PIM^c and RIM^c, the related initial margin (segregated collateral) posted and received by the bank;
- RC and CR, the reserve capital and capital at risk of the bank.

The contractually promised cash flows are supposed to be hedged out by the bank but one conservatively assumes no XVA hedge, so that the bank is left with the following trading cash flows $\mathcal{C}$ and $\mathcal{F}$ (cf. (1.45) and see [Crépey, 2022](Lemmas 5.1 and 5.2) for detailed derivations of analogous equations in a slightly simplified setup):

- The (counterparty) credit cash flows

$$\mathrm{d}\mathcal{C}_t = \tag{1.58}$$
$$\sum_{c;\tau_c\le\tau^\delta}(1-R_c)\left((P^c+\mathcal{P}^c)_{\tau_c^\delta\wedge\tau^\delta}-(\mathcal{P}^c+\mathrm{VM}^c+\mathrm{RIM}^c)_{(\tau_c\wedge\tau)-}\right)^+\boldsymbol{\delta}_{\tau_c^\delta\wedge\tau^\delta}(dt) \tag{1.59}$$
$$-(1-R)\sum_{c;\tau\le\tau_c^\delta}\left((P^c+\mathcal{P}^c)_{\tau^\delta\wedge\tau_c^\delta}-(\mathcal{P}^c+\mathrm{VM}^c-\mathrm{PIM}^c)_{(\tau\wedge\tau_c)-}\right)^-\boldsymbol{\delta}_{\tau^\delta\wedge\tau_c^\delta}(dt); \tag{1.60}$$

- The (risky) funding cash flows

$$\mathrm{d}\mathcal{F}_t = J_t\,\lambda_t\left(\sum_c J^c\,(P^c-\mathrm{VM}^c)-\mathrm{RC}-\mathrm{CR}\right)_t^+ dt \tag{1.61}$$
$$-(1-R)\left(\sum_c J^c\,(P^c-\mathrm{VM}^c)-\mathrm{RC}-\mathrm{CR}\right)_{\tau-}^+\boldsymbol{\delta}_\tau(dt) \tag{1.62}$$
$$+J_t\,\tilde{\lambda}_t\sum_c J_t^c\,\mathrm{PIM}_t^c\,dt-(1-\tilde{R})\sum_c J_{\tau-}^c\,\mathrm{PIM}_{\tau-}^c\,\boldsymbol{\delta}_\tau(dt), \tag{1.63}$$

  where the RC and CR terms account for the fungibility of reserve capital and capital at risk with variation margin.

1.6.2 Valuation

Here (as in our numerics) we distinguish between a (strict) FVA, in the strict sense of the cost of raising variation margin, and an MVA for the cost of raising initial margin (see Remark 1.1). The (other than K)VA equations are then

$$\mathrm{RC}=\mathrm{CA}=\mathrm{CVA}+\mathrm{FVA}+\mathrm{MVA} \tag{1.64}$$

the so-called "contra-assets valuation" sourced from the clients and deposited in the reserve capital account of the bank, where, for $t<\tau$,

$$\mathrm{CVA}_t=\mathbb{E}_t\sum_{t<\tau_c^\delta}(1-R_c)\left((P^c+\mathcal{P}^c)_{\tau_c^\delta}-(\mathcal{P}^c+\mathrm{VM}^c+\mathrm{RIM}^c)_{\tau_c-}\right)^+ \tag{1.65}$$

$$\mathrm{FVA}_t=\mathbb{E}_t\int_t^T\lambda_s\left(\sum_c J^c\,(P^c-\mathrm{VM}^c)-\mathrm{CA}-\mathrm{CR}\right)_s^+ ds \tag{1.66}$$

$$\mathrm{MVA}_t=\mathbb{E}_t\int_t^T\lambda_s\sum_c J_s^c\,\mathrm{PIM}_s^c\,ds. \tag{1.67}$$

The corresponding trading loss and profit process L of the bank is such that

$$\begin{aligned}
&L_0 = 0 \text{ and, for } t < \tau,\\
&d\,L_t = \sum_c (1-R_c)\left((P^c+\mathcal{P}^c)_{\tau_c^\delta} - (\mathcal{P}^c+\mathrm{VM}^c+\mathrm{RIM}^c)_{\tau_c-}\right)^+ \boldsymbol{\delta}_{\tau_c^\delta}(d\,t)\\
&+\lambda_t\left(\sum_c J^c\,(P^c-\mathrm{VM}^c) - \mathrm{CA} - \mathrm{CR}\right)^+_t d\,t\\
&+\lambda_t \sum_c J^c_t\,\mathrm{PIM}^c_t\,d\,t\\
&+\mathrm{dCA}_t,
\end{aligned} \tag{1.68}$$

so that L is a $\mathbb{Q}$ martingale, hence (by Lemma 1.18) L° is a $\mathbb{Q}^\star$ martingale.

By the same rationale as Definitions 1.13 and 1.14 in the static setup:

Definition 1.27. EC_t *is the time-t conditional 97.5% expected shortfall of* $(L^\circ_{t+1} - L^\circ_t)$ *under* $\mathbb{Q}$.

Given a positive target hurdle rate h:

Definition 1.28. *We set*

$$\mathrm{CR} = \max(\mathrm{EC}, \mathrm{KVA}), \tag{1.69}$$

for a KVA *process such that, for* $t < \tau$,

$$\mathrm{KVA}_t = \mathbb{E}_t\left[\int_t^T h\,(\mathrm{CR}_s - \mathrm{KVA}_s)\,d\,s\right]. \tag{1.70}$$

Hence, for $t < \tau$,

$$\mathrm{KVA}_t = \mathbb{E}_t\left[\int_t^T h\,e^{-h(s-t)}\mathrm{CR}_s d\,s\right] \tag{1.71}$$

$$= \mathbb{E}_t\left[\int_t^T h\,e^{-h(s-t)}\max(\mathrm{EC}_s, \mathrm{KVA}_s)\,d\,s\right]. \tag{1.72}$$

The next-to-last identity is the continuous-time analog of the risk margin formula under the Swiss solvency test cost of capital methodology: see [Swiss Federal Office of Private Insurance, 2006](Section 6, middle of page 86 and top of page 88).

1.6.3 The XVA Equations are Well-Posed

In view of (1.64), the second line in (1.65) is in fact an FVA *equation*. Likewise, the second line in (1.71) is a KVA equation. Moreover, as capital at risk is fungible with variation margin (cf. Section 1.3.4), i.e. in consideration of the CR term in (1.65)-(1.68), where $\mathrm{CR} = \max(\mathrm{EC}, \mathrm{KVA})$, we actually deal with an $(\mathrm{FVA}, \mathrm{KVA})$ *system*, and even, as EC depends on L (cf. Definition 1.27), with a forward backward system for the forward loss process L and the backward pair $(\mathrm{FVA}, \mathrm{KVA})$.

However, as in the refined static setup of Section 1.3.4, the coupling between (FVA, KVA) and L can be disentangled by the following Picard iteration:

- Let CVA and MVA be as in (1.65), $L^{(0)} = \mathrm{KVA}^{(0)} = 0$, and , for $t < \tau$,

$$\mathrm{FVA}^{(0)}_t = \mathbb{E}_t \int_t^T \lambda_s\left(\sum_c J^c\,(P^c - \mathrm{VM}^c) - \mathrm{CA}^{(0)}\right)^+_s d\,s, \tag{1.73}$$

where $\mathrm{CA}^{(0)}=\mathrm{CVA}+\mathrm{FVA}^{(0)}+\mathrm{MVA}$;

- For $k\geq 1$, writing explicitly $\mathrm{EC}=\mathrm{EC}(L)$ to emphasize the dependence of EC on L, let $L_0^{(k)}=0$ and, for $t<\tau$,

$$\begin{aligned}
d\,L_t^{(k)} &= \sum_c (1-R_c)\left((P^c+\mathcal{P}^c)_{\tau_c^\delta}-(\mathcal{P}^c+\mathrm{VM}^c+\mathrm{RIM}^c)_{\tau_c-}\right)^+ \boldsymbol{\delta}_{\tau_c^\delta}(d\,t)\\
&\quad +\lambda_t\Big(\sum_c J^c\,(P^c-\mathrm{VM}^c)-\mathrm{CA}^{(k-1)}-\max\left(\mathrm{EC}(L^{(k-1)}),\mathrm{KVA}^{(k-1)}\right)\Big)^+_t\,dt\\
&\quad +\lambda_t\sum_c J_t^c\,\mathrm{PIM}_t^c\,d\,t+\mathrm{dCA}_t^{(k-1)},\\
\mathrm{KVA}_t^{(k)} &= h\mathbb{E}_t\int_t^T e^{-h(s-t)}\max\left(\mathrm{EC}_s(L^{(k)}),\mathrm{KVA}_s^{(k)}\right)d\,s,\\
\mathrm{CA}_t^{(k)} &= \mathrm{CVA}_t+\mathrm{FVA}_t^{(k)}+\mathrm{MVA}_t\\
\mathrm{FVA}_t^{(k)} &= \mathbb{E}_t\int_t^T\lambda_s\Big(\sum_c J^c\,(P^c-\mathrm{VM}^c)-\mathrm{CA}^{(k)}-\max\left(\mathrm{EC}(L^{(k)}),\mathrm{KVA}^{(k)}\right)\Big)^+_s\,d\,s.
\end{aligned}\tag{1.74}$$

Theorem 4.1 in [Crépey et al., 2020] Assuming square integrable data, the XVA equations are well-posed within square integrable solution (including when one accounts for the fact that capital at risk can be used for funding variation margin). Moreover, the above Picard iteration converges to the unique square integrable solution of the XVA equations.

1.6.4 Collateralization Schemes

We denote by $\Delta_t^c=\mathcal{P}_t^c-\mathcal{P}_{(t-\delta)-}^c$ the cumulative contractual cash flows with the counterparty c accumulated over a past period of length δ. In our case study, we consider both "no CSA" netting sets c, with $\mathrm{VM}=\mathrm{RIM}=\mathrm{PIM}=0$, and "(VM/IM) CSA" netting sets c, with $\mathrm{VM}_t^c=P_t^c$ and, for $t\leq\tau_c$,

$$\mathrm{RIM}_t^c=\mathbb{V}a\mathbb{R}_t((P_{t^\delta}^c+\Delta_{t^\delta}^c)-P_t^c),\,\beta_t\,\mathrm{PIM}_t^c=\mathbb{V}a\mathbb{R}_t(-(P_{t^\delta}^c+\Delta_{t^\delta}^c)+P_t^c),\tag{1.75}$$

for some PIM and RIM quantile levels a_{pim} and a_{rim} (and $t^\delta=t+\delta$).

The following result can be derived by similar computations as the ones in [Albanese et al., 2020](Section A).

Proposition 1.29. *In a common shock default model of the counterparties and the bank itself (see the beginning of Section 1.5), with pre-default intensity processes γ^c of the counterparties and γ of the bank, then* $\mathrm{CVA}=\mathrm{CVA}^{nocsa}+\mathrm{CVA}^{csa}$, *where, for $t<\tau$,*

$$\begin{aligned}
\mathrm{CVA}_t^{nocsa} &= \sum_{c\ \ nocsa}\mathbb{1}_{t<\tau_c}(1-R_c)\mathbb{E}_t\int_t^T(P_{s^\delta}^c+\Delta_{s^\delta}^c)^+\,\gamma_s^c\,e^{-\int_t^s\gamma_u^c du}\,d\,s\\
&\quad +\sum_{c\ \ nosca}\mathbb{1}_{\tau_c<t<\tau_c^\delta}(1-R_c)\mathbb{E}_t(P_{\tau_c^\delta}^c+\Delta_{\tau_c^\delta}^c)^+,
\end{aligned}\tag{1.76}$$

$$\begin{aligned}
\mathrm{CVA}_t^{csa} &= \sum_{c\ \ csa}\mathbb{1}_{t<\tau_c}(1-R_c)\,(1-a_{\mathrm{rim}})\times\\
&\quad \mathbb{E}_t\int_t^T(\mathbb{ES}_s-\mathbb{V}a\mathbb{R}_s)\left((P_{s^\delta}^c+\Delta_{s^\delta}^c)-P_s^c\right)\gamma_s^c\,e^{-\int_t^s\gamma_u^c du}\,d\,s\\
&\quad +\sum_{c\ \ csa}\mathbb{1}_{\tau_c<t<\tau_c^\delta}(1-R_c)\mathbb{E}_t\big((P_{\tau_c^\delta}^c+\Delta_{\tau_c^\delta}^c)-(P_{\tau_c}^c+\mathrm{RIM}_{\tau_c}^c)\big)^+,
\end{aligned}\tag{1.77}$$

where $(\mathbb{ES}_s - \mathbb{V}a\mathbb{R}_s)$ *in (1.77) is computed at the* a_{rim} *confidence level. Assuming its posted initial margin borrowed unsecured by the bank, then* $\text{MVA} = \text{MVA}^{csa}$*, where, for* $t < \tau$*,*

$$\text{MVA}_t^{csa} = \sum_{c \quad csa} J_t^c \, \mathbb{E}_t \int_t^T (1-R)\gamma_s \, \text{PIM}_s^c \, e^{-\int_t^s \gamma_u^c du} \, d\, s. \tag{1.78}$$

Chapter 2

Pathwise CVA Regressions With Over-simulated Defaults

This chapter, accepted for publication in Mathematical Finance, was co-authored with Lokman A. Abbas-Turki and Stéphane Crépey.

We consider the computation by simulation and neural net regression of conditional expectations, or more general elicitable statistics, of functionals of processes (X, Y). Here an exogenous component Y (Markov by itself) is time-consuming to simulate, while the endogenous component X (jointly Markov with Y) is quick to simulate given Y, but is responsible for most of the variance of the simulated payoff. To address the related variance issue, we introduce a conditionally independent, hierarchical simulation scheme, where several paths of X are simulated for each simulated path of Y. We analyze the statistical convergence of the regression learning scheme based on such block-dependent data. We derive heuristics on the number of paths that should be simulated. The resulting algorithm is optimally implemented on a graphics processing unit (GPU) combining Python/CUDA and learning with PyTorch. A CVA benchmarking case study of the method with a reference nested Monte Carlo approach shows that the hierarchical simulation technique is key to the success of the learning approach.

An optimized GPU implementation of our hierarchical simulation and regression learning scheme, as well as single-file python notebook demo for our CVA use case, are available at `https://github.com/BouazzaSE/NeuralXVA`.

2.1 Introduction

Greensill defaulted on March 8, 2021, a collapse estimated by British parliamentarians to trigger a cost for UK taxpayers of up to £5bn[2.1]. Greensill fell short of capital because they lent to Gupta against future invoices which then did not materialize. A projection of Greensill's capital requirements including the tail risk related to Gupta's default would have highlighted a sizable concentrated and unsecured credit risk to a junk-rated counterparty. This example emphasizes the interest of default risk simulations, as opposed to credit spread simulations simply, as typically done in the industry for the sake of simplicity. Another, related example is the path-wise XVA regressions of Albanese et al., 2021, which also require a hybrid market and credit setup, where the actual defaults of the clients of the bank are simulated.

2.1. cf. https://www.theguardian.com/business/2021/apr/28/greensill-collapse-could-cost-uk-taxpayer-up-to-5bn-mps-told.

However, to obtain the required level of accuracy in a simulation setup with defaults, one needs a very large number of simulations—100 times more, say, than in a standard mark-to-market simulation without simulation of the defaults, where 100 is in reference to credit spreads that would be of the order of 1%. A factor 100 is not necessarily a sizable amount as far as the simulation of the risk factors is involved. But it also means that the mark-to-market cube of path-wise prices of all trades of the bank becomes 100 times larger. When applied at the level of a realistic banking portfolio, this becomes prohibitive in terms of both computation time and memory occupancy.

To overcome this problem, we introduce in this paper an acceleration technique for the computation by simulation and regression of conditional expectations of functions ξ of Markov pairs (X, Y), where Y is an exogenous component, Markov by itself, whose simulation is time-consuming, while the endogenous component X (jointly Markov with Y) is quick to simulate given Y, but also responsible for most of the variance of the simulated payoff. The idea, which we call the hierarchical simulation setup, is then to draw an optimized number of realizations of X conditional on each simulation of Y. For example, in the above-mentioned XVA regression framework, we simulate a few hundred paths of client defaults conditionally on each mark-to-market path. Proceeding in this way, the computational burden of the mark-to-market cube is not amplified by the simulation of the client defaults. We demonstrate, both mathematically and empirically, that the lack of independence of the ensuing simulation setup is not detrimental to the quality of the ensuing learner, i.e. (in the above case) of the regressions of the XVA layers built over the mark-to-market cube and defaults scenarios.

Note that the use of regression-based Monte Carlo simulations for XVA computations is not new. It was already presented in Cesari et al., 2010 as a key CVA computational paradigm, intended to avoid nested Monte Carlo. However, from traditional XVA computations to [Huge and Savine, 2020], the regressions are used for computing the mark-to-market cube of the prices of all the contracts of the bank with all its clients (or netting sets) at all times of a simulation time-grid, out of which the CVA of the bank at time 0 (and only it) is obtained by integration of the so-called expected positive exposure relative to each netting set against the credit curve of the corresponding client, and summation over netting sets. By contrast, in this paper, we aim at learning the CVA as a process, i.e. at every node of a simulation for all risk factors, based on a mark-to-market cube computed by model analytics at the forward simulation stage. Regressions for the mark-to-market of derivatives are typically multiple standard parametric regressions in a diffusive and low-dimensional setup, as opposed to the hybrid and high-dimensional neural net regressions that we handle in this paper. Recently, Gnoatto et al., 2021 deep-hedge and learn the CVA and the FVA, but this is again in a purely diffusive setup, after the default of the bank and its (assumed single) counterparty have been eliminated from the model by the reduction of filtration technique of Crépey and Song, 2015. We believe that this technique of reduction of filtration is not extensible to the realistic case of a bank involved in transactions with several (in fact, many, e.g. several thousands of) clients, the default times of which enter the ensuing FVA and KVA equations in a nonlinear fashion, so that there is no other choice but simulating these defaults and including them in the regressions—but this requires special care, which is the topic of our work.

2.1.1 Outline

In Section 2.2, we introduce a neural net learning framework for conditional expectations, iterated in time as they appear naturally in dynamic pricing problems, taking into account the dynamics of the problem by means of a backward algorithm. In Section 2.4,

we introduce a CVA case study. In Section 2.3, we identify the variance issue at hand and we propose a hierarchical simulation approach to address it. We establish the benefit of this approach mathematically by providing associated generalization bounds. Section 2.4.4 illustrates it numerically.

Note that, although our CVA case study only covers quadratic risk minimization (for benchmarking purposes), the approach and the proofs of this paper are valid for more general loss functions and apply to the learning of any elicitable statistics. In particular, via the Rockafellar and Uryasev, 2000 representation of value-at-risk and expected shortfall of a given loss (random variable) in terms of "far out-of-the-money call options" on that loss, our hierarchical simulation approach is also relevant for learning value-at-risk and expected shortfall in hybrid mark-to-market and default simulation setups. Such an approach is even particularly relevant in these cases, where the fact that X is responsible for most of the variance of the payoff is then intrinsic to the far out-of-the-money feature of the corresponding "option".

2.2 Neural Regression Setup

A reference probability space, with corresponding probability measure and expectation denoted by $\mathbb{Q}$ and $\mathbb{E}$, is fixed throughout the paper. The state spaces of X and Y are taken as $\mathbb{R}^p$ and $\mathbb{R}^q$, for some positive integers p and q. In the (default risk) case of a Markov chain like component X, referred to hereafter as the Markov chain X case (but with transition intensities modulated by Y), we assume, without loss of generality in this case, that X evolves on the vertices $\{0,1\}^p$ of the unit cube in $\mathbb{R}^p$. We take the problem after discretisation of time (if the latter was continuous in the first place), for a time step set to one year for ease of notation.

We then consider $(X_i)_{0\le i\le n}$ and $(Y_i)_{0\le i\le n}$ as discrete-time processes on the time grid. Our goal is to estimate, for every i, conditional expectations of the form

$$\Pi_i = \mathbb{E}[\xi_{i,n}|X_i, Y_i], \tag{2.1}$$

where

$$\xi_{i,n} := f_i(X_i, \dots, X_n, Y_i, \dots, Y_n). \tag{2.2}$$

Here f_i is a measurable real function such that $\xi_{i,n}$ is a square-integrable random variable.

Conditional expectations such as (2.1) can be estimated via linear regression using a finite sample. This is ubiquitous in quantitative finance since the Bermudan Monte Carlo papers of Tsitsiklis and Van Roy, 2001 and Longstaff and Schwartz, 2001. In order to estimate the conditional expectation in (2.1), one draws i.i.d. samples $\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota)\}_{\iota\in\mathcal{I}}$ of $(X_i, Y_i, \xi_{i,n})$ where $\mathcal{I}$ is a finite set of indices. Then, given a feature map $\phi\colon \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}^m$ (for some positive integer m), one linearly regresses $\{\xi_{i,n}^\iota\}_{\iota\in\mathcal{I}}$ against $\{\phi(X_i^\iota, Y_i^\iota)\}_{\iota\in\mathcal{I}}$, solving for

$$\hat{w}_i \in \operatorname*{argmin}_{w_i\in\mathbb{R}^m} \sum_{\iota\in\mathcal{I}} (\xi_{i,n}^\iota - w_i^\top \phi(X_i^\iota, Y_i^\iota))^2. \tag{2.3}$$

One then uses $\hat{w}_i^\top \phi(X_i, Y_i)$ as an approximation for Π_i.

The above procedure is justified by the characterization, in the square integrable case, of conditional expectations as orthogonal projections, i.e.

$$\mathbb{E}[\xi_{i,n}|X_i, Y_i] = \varphi_i^\star(X_i, Y_i) \quad \text{a.s.},$$

where, denoting by $\mathcal{B}(E)$ the set of Borel measurable real functions on a metric space E,

$$\varphi_i^\star \in \underset{\varphi_i \in \mathcal{B}(\mathbb{R}^p \times \mathbb{R}^q)}{\operatorname{argmin}} \mathbb{E}[(\xi_{i,n} - \varphi_i(X_i, Y_i))^2]. \tag{2.4}$$

One recovers the linear regression formulation (2.3) by approximating the expectation by an empirical mean and restricting the search space to the functions of the form $\mathbb{R}^p \times \mathbb{R}^q \ni (x, y) \mapsto w_i^\top \phi(x, y)$, where $w_i \in \mathbb{R}^m$.

2.2.1 Neural Net Parameterization

Linear regression by means of a priori, explicit factors has a reasonable chance of success when $\varphi_i^\star$ is simple enough and the feature mapping ϕ can be judiciously chosen, usually from expert knowledge. This is however not always the case, e.g. when considering portfolio-wide XVA metrics, which exhibit non-trivial dependencies on the many risk factors being regressed against. It is then impossible to manually devise a satisfactory feature mapping ϕ.

Figure 2.1 shows how a linear regression with the raw risk factors as features fails for even a simple portfolio comprised of a call option, while the neural net estimator almost matches with the nested Monte Carlo estimator (see Sections 2.4 and 2.4.4 for more numerical details).

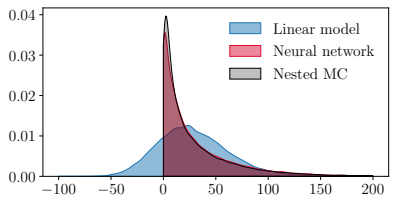

Figure 2.1. Density plot of the CVA of a vanilla call, at mid-life of the option.

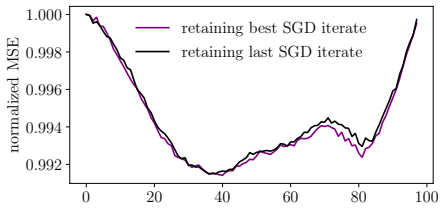

Figure 2.2. Out-of-sample MSEs against labels $\xi_{i,n}$ at different time-steps divided by the variance of the labels.

In the Markov chain X case, we face the additional peculiarity of a hybrid regression setting, in view of the discrete and continuous natures of the X and Y model components.

Neural networks [Bengio et al., 2016] propose an alternative way to parameterize and learn the feature map. Let $\mathcal{NN}_{p+q,h,u,\varsigma}$ denote the set of functions of the form

$$\mathbb{R}^{p+q} \ni z \mapsto \zeta(z; W^{[h+1]}, \dots, W^{[1]}, b^{[h+1]}, \dots, b^{[1]}) = \zeta^{[h+1]}(z; W, b)$$

where $W^{[h+1]} \in \mathbb{R}^{1\times u}, \dots, W^{[\ell]} \in \mathbb{R}^{u\times u}, \dots, W^{[1]} \in \mathbb{R}^{u\times(p+q)}$ are the weight matrices, $b^{[h+1]} \in \mathbb{R}, \dots, b^{[\ell]} \in \mathbb{R}^u, \dots, b^{[1]} \in \mathbb{R}^u$ are the bias offsets, W and b are the respective concatenations of the $W^{[\ell]}$ and of the $b^{[\ell]}$, ς is an element-wise scalar nonlinearity and, for every $z \in \mathbb{R}^{p+q}$,

$$\begin{aligned}
&\zeta^{[0]}(z; W, b) = z \\
&\zeta^{[\ell]}(z; W, b) = \varsigma\left(W^{[\ell]}\, \zeta^{[\ell-1]}(z; W, b) + b^{[\ell]}\right), \quad \ell = 1, \dots, h \\
&\zeta^{[h+1]}(z; W, b) = W^{[h+1]}\, \zeta^{(h)}(z; W, b) + b^{[h+1]}.
\end{aligned}$$

The function $z \mapsto \zeta^{[h+1]}(z; W, b)$ can be seen as a nonlinear feature mapping from $\mathbb{R}^{p+q}$ to $\mathbb{R}^u$, parameterized by $W^{[h+1]}, \dots, W^{[1]}, b^{[h+1]}, \dots, b^{[1]}$ (for a given activation function ς). On top of the set $\mathcal{NN}_{p+q,h,u,\varsigma}$ of real-valued neural networks taking inputs from $\mathbb{R}^{p+q}$, with h hidden layers, u units per hidden layer, and ς as the activation function, we also define

$$\mathcal{NN}^+_{p+q,h,u,\varsigma} := \{\mathbb{R}^{p+q} \ni z \mapsto (f(z))^+ + \mu,\, f \in \mathcal{NN}_{p+q,h,u,\varsigma},\, \mu \in \mathbb{R}\}. \tag{2.5}$$

This specification ensures positivity of the output when the additive constant μ is non-negative and is useful for learning positive (e.g. XVA) functions. The additive constant μ is introduced in order to improve the fit of the first moment and hence reduce the bias.

In what follows, we identify $\mathbb{R}^{p+q}$ with $\mathbb{R}^p \times \mathbb{R}^q$ and write $\phi(z)$ or $\phi(x, y)$ interchangeably, where z is the concatenation of x and y, for every function ϕ defined over $\mathbb{R}^{p+q}$ or $\mathbb{R}^p \times \mathbb{R}^q$, and for every $x \in \mathbb{R}^p$ and $y \in \mathbb{R}^q$.

2.2.2 Local Training Algorithm

Learning the conditional expectation (2.1) in a positive neural net search space consists in applying the same empirical risk minimization (2.3) approximation as in linear regression, using this time $\mathcal{NN}^+_{p+q,h,u,\varsigma}$ as the search space, i.e. solving for

$$\hat{\varphi}_i \in \operatorname*{argmin}_{\varphi_i \in \mathcal{NN}^+_{p+q,h,u,\varsigma}} \sum_{\iota \in \mathcal{I}} (\xi^\iota_{i,n} - \varphi_i(X^\iota_i, Y^\iota_i))^2. \tag{2.6}$$

This is achieved by using an iterative gradient-based optimization algorithm, which we will assume to be mini-batch stochastic gradient descent.

In the context of learning a positive output (e.g. an XVA), the addition of a ReLU activation $(\cdot)^+$ at the output layer in (2.5) can jeopardize the learning as the gradient may vanish at a certain SGD iteration and the parameters are then frozen irrespective of the number of subsequent iterations. Thus, for more stability of the learning procedure, we first perform the first half of SGD steps on the network without the ReLU at the output layer. Then, still without the ReLU, we fine-tune the weights of the output layer by optimizing with respect to those weights only (freezing the weights of the hidden layers), which can be done in closed form in the case of quadratic risk minimization.

Remark 2.1. This step isn't achievable in closed-form in the case of, for example, quantile regression[2.2]. However, even in this case, the optimization problem is still convex and as such easier to solve numerically.

Finally, we restore the ReLU at the output layer and we finish the last half of the SGD iterations.

We also chose to retain the best set of parameters among those explored during the SGD iterations. Figure 2.2 shows the corresponding improvement in generalization when applied in the context of the case study of Sections 2.4 and 2.4.4.

The ensuing learning scheme is detailed in Algorithm 2.1. Note that we presented vanilla SGD iterations only for the sake of simplicity. In practice, accelerated SGD methods like Adam ([Kingma and Ba, 2014]) are used instead.

2.2. cf. the last paragraph of Section 2.1.

Algorithm 2.1
Baseline learning scheme for training at a given time-step i

name: BaseAlg
input: $\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota), \iota \in \mathcal{I}\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E \in \mathbb{N}^\star$, a learning rate $\eta > 0$, initial values for the network parameters W, b and μ
output: Trained parameters W_{best}, b_{best} and μ_{best}
define

$$\mathcal{L}(W, b, \mu, \text{batch}, \text{pos}) = \begin{cases} \frac{1}{|\text{batch}|} \sum_{\iota \in \text{batch}} (\zeta^{[h+1]}(X_i^\iota, Y_i^\iota; W, b) + \mu - \xi_{i,n}^\iota)^2 & \text{if pos} = 0 \\ \frac{1}{|\text{batch}|} \sum_{\iota \in \text{batch}} ((\zeta^{[h+1]}(X_i^\iota, Y_i^\iota; W, b))^+ + \mu - \xi_{i,n}^\iota)^2 & \text{if pos} = 1 \end{cases}$$

$\mathcal{L}_{\text{best}} \leftarrow \infty$, pos$\leftarrow$0
// loop over epochs
for epoch=$1, \ldots, E$ **do**
  // loop over batches
  for batch$\in \mathcal{B}$ **do**
    for $\ell = 1, \ldots, h+1$ **do**
      $W^{[\ell]} \leftarrow W^{[\ell]} - \eta \nabla_{W^{[\ell]}} \mathcal{L}(W, b, \mu, \text{batch}, \text{pos})$
      $b^{[\ell]} \leftarrow b^{[\ell]} - \eta \nabla_{b^{[\ell]}} \mathcal{L}(W, b, \mu, \text{batch}, \text{pos})$
    end
    $\mu \leftarrow \mu - \eta \partial_\mu \mathcal{L}(W, b, \mu, \text{batch}, \text{pos})$
  end
  if epoch=$\lfloor \frac{E}{2} \rfloor$ **then**
    // tune weights of last layer
    $(W^{[h+1]}, b^{[h+1]}) \leftarrow \text{argmin}_{\tilde{W}^{[h+1]}, \tilde{b}^{[h+1]}} \mathcal{L}(\{W^{[0]}, \ldots, W^{[h]}, \tilde{W}^{[h+1]}\}, \{b^{[0]}, \ldots, b^{[h]}, \tilde{b}^{[h+1]}\}, \mu, obs, 0)$
    pos$\leftarrow$1
  end
  if $\mathcal{L}(W, b, \mu, \mathcal{I}, 1) < \mathcal{L}_{\text{best}}$ **then**
    // keep track of best parameters
    $\mathcal{L}_{\text{best}} \leftarrow \mathcal{L}(W, b, \mu, obs, 1)$
    $W_{\text{best}} \leftarrow W$
    $b_{\text{best}} \leftarrow b$
    $\mu_{\text{best}} \leftarrow \mu$
  end
end

2.2.3 Backward Learning

In the setup of the path-wise pricing problem (2.1), at each pricing time i, a separate learning problem is solved by Algorithm 2.1. Since the algorithm returns for each problem a local minimum, it is possible to end up with an approximation of the pricing function $\mathbb{E}[\xi_{i,n} | X_i = x, Y_i = y]$ (cf. (2.1)) with noisy paths (i.e. with respect to time i) if the local minima are not close to each other, even for fixed x and y. Yet, for two consecutive time-steps i and $i+1$, the learning problems are similar. One possible refinement is, after having learned $\xi_{i+1,n}$, to initialize the parameters of the network at time i with the parameters of the network trained at time $i+1$. This not only smoothes the results across regression times, but also accelerates convergence.

We obtain an algorithm which starts the learnings at time step n and, proceeding backward in time until time step 1, reuses each time the previous solution as an initialization for the next learning. The ensuing backward learning scheme is detailed in Algorithm 2.2. This process of reusing knowledge from a different but related learning task can be seen as a form of transfer learning [Pan and Yang, 2009; Bozinovski, 2020].

Algorithm 2.2
Backward learning scheme

input: $\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota), \iota \in \mathcal{I}, 1 \le i \le n\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E \in \mathbb{N}^\star$, a learning rate $\eta > 0$
output: $\hat{\varphi}_1, \dots, \hat{\varphi}_n$
initialize parameters W_{n+1}, b_{n+1} and μ_{n+1} of the network at terminal time-step n
`// loop backwards over the time steps`
for $i = n, \dots, 1$ **do**
  $W_i, b_i, \mu_i \leftarrow \text{BaseAlg}(\{(X_i^\iota, Y_i^\iota, \xi_{i,n}^\iota), \iota \in \mathcal{I}\}, \mathcal{B}, E, \eta, W_{i+1}, b_{i+1}, \mu_{i+1})$
  $\hat{\varphi}_i \leftarrow \{x \mapsto \zeta^{[h+1]}(x, y; W_i, b_i) + \mu_i\}$
end

Remark 2.2. A variation on the above would be forward learning. We favor the backward learning scheme because it is the only one that is amenable to more general backward stochastic differential equations, such as the equations for the FVA and the KVA in [Crépey et al., 2020]. In addition, in these XVA applications, the labels/features corresponding to times i close to the final maturity n of the portfolio have a lower/higher variance. Hence the training task corresponding to a higher i is easier.

2.2.4 Separable Case

Next we present a fine-tuning which is applicable when $\xi_{i,n} = \sum_{j=1}^p \xi_{i,n}^{(j)}$, where, for every $1 \le j \le p$, denoting by $x^{(j)}$ the j^{th} component of $x \in \mathbb{R}^p$, one has (cf. (2.2))

$$\xi_{i,n}^{(j)} := f_i^{(j)}(X_i^{(j)}, \dots, X_n^{(j)}, Y_i, \dots, Y_n)$$

for some real function $f_i^{(j)}$ such that $\xi_{i,n}^{(j)}$ is square-integrable. Then

$$\mathbb{E}[\xi_{i,n}^{(j)} | X_i, Y_i] = \mathbb{E}[\xi_{i,n}^{(j)} | X_i^{(j)}, Y_i] = \Pi_i^{(j)},$$

which can be learned separately for each coordinate j.

In the Markov chain case with state space $\{0,1\}^p$ of X, we can write

$$\Pi_i^{(j)} = \mathbb{E}[\xi_{i,n}^{(j)} | \{X_i^{(j)} = 1\}, Y_i] X_i^{(j)} + \mathbb{E}[\xi_{i,n}^{(j)} | \{X_i^{(j)} = 0\}, Y_i] \, (1 - X_i^{(j)}). \tag{2.7}$$

Thus, for every i we have two sub-learning problems, respectively conditional on $\{X_i^{(j)} = 1\}$ and $\{X_i^{(j)} = 0\}$, and the feature $X_i^{(j)}$ is no longer needed in the regressions. Algorithms 2.1 and 2.2 can be easily adapted to this setting by averaging over the respective samples where $X_i^{(j)} = 1$ and 0 (instead of averaging over the whole dataset as before). A requirement is to have enough samples for both events, but this can be facilitated by the approach presented in Section 2.3.

Separability comes in handy when learning for example a CVA or an MVA for each counterparty of a bank (whether default indicator based as in (2.20) or default intensity based as in (2.22)). However, it is not applicable to FVA computations and KVA computations, which can only be addressed at the level of the overall portfolio of the bank [Albanese et al., 2021].

2.2.5 A Posteriori Twin Monte Carlo Validation Procedure

As part of the validation of our approach, we computed a benchmark estimator and compared the learning approach against it by computing L^2 error estimates. In fact, one can compute such L^2 error estimates without necessarily performing a slow nested Monte Carlo benchmark run. At a given time step i, let $\xi_{i,n}^{(1)}$ and $\xi_{i,n}^{(2)}$ denote two independent copies of $\xi_{i,n}$ conditional on (X_i, Y_i)[2.3]. For any Borel function $\varphi\colon \mathbb{R}^p \times \mathbb{R}^q \to \mathbb{R}$ such that $\varphi(X_i, Y_i)$ is square integrable (e.g. a neural net estimate of $\mathbb{E}[\xi_{i,n}|X_i, Y_i]$), we have:

$$\mathbb{E}[(\varphi(X_i, Y_i) - \mathbb{E}[\xi_{i,n}|X_i, Y_i])^2] = \mathbb{E}\left[\varphi(X_i, Y_i)^2 - (\xi_{i,n}^{(1)} + \xi_{i,n}^{(2)})\,\varphi(X_i, Y_i) + \xi_{i,n}^{(1)}\,\xi_{i,n}^{(2)}\right]. \tag{2.8}$$

The equality stems from the fact that, by conditional independence,

$$\mathbb{E}[\xi_{i,n}|X_i, Y_i]^2 = \mathbb{E}[\xi_{i,n}^{(1)}|X_i, Y_i]\,\mathbb{E}[\xi_{i,n}^{(2)}|X_i, Y_i] = \mathbb{E}[\xi_{i,n}^{(1)}\,\xi_{i,n}^{(2)}|X_i, Y_i], \tag{2.9}$$

followed by an application of the tower rule. Thus, one is able to approximate the L^2 error of any estimator for the conditional expectation, without any knowledge on the latter, using only two inner paths. This can be used as a very fast validation procedure and as a safeguard in a production environment before using the learned values. A slower but more complete nested Monte Carlo approach is then only needed periodically, e.g. after significant changes in the risk factor models, or to perform more elaborate checks (e.g. tail behavior).

2.2.6 Python/CUDA Optimized Implementation Using GPU

Contrary to most use-cases of machine learning where the final product is the trained model and thus execution time is only critical during inference, in the case of learning from simulated data in pricing applications, the training process itself is part of the final product. Hence particular care is needed when writing the training procedures.

We implemented Algorithm 2.2 using Python programming with the CUDA API (Application Programming Interface). Because the considered problem involves high variances and thus requires a sufficiently large sample size, both training and inference are not easy to achieve in a reasonable execution time. First, we need to leverage the many-core parallel architecture of GPUs that involves streaming multiprocessors, which are used for the simulation, learning and inference phases. All phases are intertwined and performed inline. Hence we need to carefully optimize each part of the algorithm.

On the simulation side, due to their intrinsically parallel nature, Monte Carlo simulations easily lend themselves to parallelization on GPUs. Nevertheless, various optimizations are needed to have achieve a reasonable solution executed within a few seconds (cf. Figure 2.8 in Section 2.4.4). We chose to use Python and the CUDA kernels are compiled *just-in-time* using the module numba, which allows to dynamically generate CUDA kernels at run-time.

Regarding learning, we opted for PyTorch for its proximity to the CUDA programming model and its *just-in-time* compiler allowing for static computation graphs and automatic fusion, whenever appropriate, of the kernels associated with the PyTorch operations used by the model.

2.3. The conditional independence means that for any Borel bounded functions ϕ_1 and ϕ_2, we have $\mathbb{E}[\phi_1(\xi_{i,n}^{(1)})\,\phi_2(\xi_{i,n}^{(2)})|X_i, Y_i] = \mathbb{E}[\phi_1(\xi_{i,n}^{(1)})|X_i, Y_i]\,\mathbb{E}[\phi_2(\xi_{i,n}^{(2)})|X_i, Y_i]$.

We used most of the optimization techniques already presented in Abbas-Turki et al., 2018, except those related to regressions since these are replaced here by neural networks. We also introduced several additional optimizations, the most important one being to judiciously manage the CPU and GPU memories. A naive solution would involve the CPU/GPU virtual unified memory [NVIDIA Corporation, 2020b] and let the compiler choose. However, this usually results in sub-optimal memory accesses. Our choice rather targets an efficient use of the GPU memory space, a reduction of CPU/GPU transfer and an optimized transfer when needed. These optimizations and implementation choices are developed in the accompanying Github repository[2.4].

2.3 Hierarchical Simulation and its Analysis

Learning from defaults based on (2.23) may be challenging, even with optimized training schemes. As should always be first scrutiny with machine learning, the difficulty here comes from the data, *i.e.* from the simulation part in our case. Specifically, a large variance of the estimated population loss function can jeopardize the learning approach, which we address in what follows by a suitable hierarchical simulation approach.

2.3.1 Identification of the Variance Contributions Using Automatic Relevance Determination

In this part we show how to hierarchize the variance impact of explanatory variables using automatic relevance determination (ARD). As detailed in [Rasmussen and Williams, 2006](Sections 5.1, 5.4.3, 6.6, and 8.3.7), ARD is a Bayesian procedure for feature selection and consists in estimating the relevance of the features by maximizing a marginal likelihood. In our setup, we apply a Gaussian process regression based ARD to quantify empirically the impact of the variances of X and Y on that of ξ.

Toward this aim, we treat the vector of the financial model parameters, denoted by ν, as a latent variable endowed with some instrumental distribution.

Given ν, we sample the following time-averages of the variances of $X_1, \dots, X_n$, $Y_1, \dots, Y_n$ and $\xi_{1,n}, \dots, \xi_{n,n}$,

$$V(X|\nu) = \frac{1}{n+1} \sum_{i=0}^{n} \hat{\mathbb{E}}^{\nu}[(X_i - \hat{\mathbb{E}}^{\nu}[X_i])^2]$$

$$V(Y|\nu) = \frac{1}{n+1} \sum_{i=0}^{n} \hat{\mathbb{E}}^{\nu}[(Y_i - \hat{\mathbb{E}}^{\nu}[Y_i])^2]$$

$$V(\xi|\nu) = \frac{1}{n+1} \sum_{i=0}^{n} \hat{\mathbb{E}}^{\nu}[(\xi_{i,n} - \hat{\mathbb{E}}^{\nu}[\xi_{i,n}])^2]$$

meant componentwise in the vector cases of X and Y, where $\hat{\mathbb{E}}^{\nu}$ is an empirical average over paths sampled for a given realization ν of the financial model parameters. Then, based on a finite sample of ν and on the corresponding realizations of the triple $(V(X|\nu)$,

2.4. https://github.com/BouazzaSE/NeuralXVA.

$V(Y|\nu), V(\xi|\nu)$), we perform a Gaussian process regression [Rasmussen and Williams, 2006] of $V(\xi|\nu)$ against $V(X|\nu)$ and $V(Y|\nu)$. In this procedure we use an anisotropic kernel $\exp\left(-\sum_{j=1}^{p}\frac{(v_j-v_j')^2}{2\,\lambda_{x,j}^2}-\sum_{j=1}^{q}\frac{(w_j-w_j')^2}{2\,\lambda_{y,j}^2}\right)$, where the hyperparameters $\lambda_{x,1},\ldots,\lambda_{x,p}$ and $\lambda_{y,1},\ldots,\lambda_{y,q}$ are characteristic length-scales for the corresponding components of $V(X|\nu)$ and $V(Y|\nu)$. Maximizing the marginal likelihood on the dataset allows to recover those length-scales and these can then be interpreted as relevance estimates for our input variables. The higher the inverse length-scale gets, the more the corresponding variable influences the output (payoff variance, in our case).

The above procedure is then itself randomized, i.e. run multiple times on the restricted datasets corresponding to different sub-samplings of ν. This provides a distribution of the fitted hyper-parameters λ in the above, while being also less prone to over-fitting and local minima issues. A similar analysis was used in Bergstra and Bengio, 2012 to study the relevance of different neural network hyper-parameters with respect to the validation loss.

Figure 2.3 reveals the dominance of the impact of the variance of X on that of ξ in the context of our CVA case study of Section 2.4, here for a single client of the bank and relying on the CVA representation (2.20), where Y is the vector of mark-to-market risk factor processes and X is the default indicator process of the client.

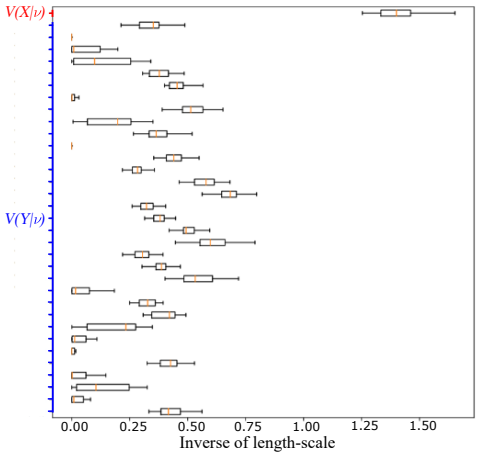

Figure 2.3. (Single client CVA (2.20)) Box-plot of the inverse length-scales obtained by randomized Gaussian process regressions of the conditional variances of the cash flows ξ against the conditional variances of the risk factors X and Y, where conditional here is in reference to the parameters of the model treated as a random vector with a postulated distribution.

2.3.2 Learning on Hierarchically Simulated Paths

If X contributes more to the variance of ξ than Y, then, in order to deal with the resulting variance issue regarding the associated simulation/learning scheme, an idea is to simulate more realizations of X than Y, even if this means giving up the independence of the simulation setup. More precisely, we simulate M i.i.d paths $Y^1,\ldots,Y^M$ of Y and, for every $k\in\{1,\ldots,M\}$ and $i\in\{1,\ldots n\}$, we simulate N i.i.d realizations $X_i^{k,1},\ldots,X_i^{k,N}$ of X_i conditional on Y^k. For every i, this yields to a sample $(X_i^{k,l},Y_i^k,\xi_{i,n}^{k,l})^k\in\{1,\ldots,M\}$, $l\in\{1,\ldots,N\}$ of $(X_i,Y_i,\xi_{i,n})$ of size MN, where, within each block k, independence between the $X^{k,l}$ only holds conditionally on Y^k.

Algorithm 2.2 is then run on the resulting hierarchically simulated dataset by taking $\mathcal{I} = \{1,\dots,M\} \times \{1,\dots,N\}$, with by convention $Y^{k,l} = Y^k$ for all l. For implementation efficiency reasons pertaining to memory contiguity, the set of indices of the $\imath$-th batch, with $1 \le \imath \le |\mathcal{B}|$, is chosen to be

$$\{(k,l) \in \{1,\dots,M\} \times \{1,\dots,N\} : (\imath-1)\frac{|\mathcal{I}|}{|\mathcal{B}|} \le (l-1)N + (k-1) + 1 < \imath\frac{|\mathcal{I}|}{|\mathcal{B}|}\}.$$

Hierarchical simulation in the above sense can be thought of as a form of data augmentation procedure (see e.g. [Shorten and Khoshgoftaar, 2019]), but in a simulation setup where one knows how to generate the data perfectly. In this framework, the main question is then to which extent one should augment the data, i.e. the choice of the hierarchical simulation parameters M and N, which is the focus of the sequel of this section.

Remark 2.3. Hierarchical simulation is different in nature from importance sampling that favors particular events, e.g., in a credit risk setup, default versus survival (see e.g. [Carmona and Crépey, 2010]). In an XVA setup, however, some metrics, like the CVA, need default events for being properly estimated, whereas others, like the FVA, require survival events. Hence what one needs is richness regarding both default and survival events, which is precisely what hierarchical simulation provides.

2.3.3 Choosing the Hierarchical Simulation Factor

Assume that simulating Y_i costs P times more than simulating X_i given a path $\{Y_j\}_{j \le i}$ in terms of computation time. The hierarchical simulation factor N can be chosen so as to minimize the variance ($\mathbb{V}\mathrm{ar}$) of the loss $\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} g_i(\theta, X_i^{k,l},\dots,X_n^{k,l},Y_i^k,\dots,Y_n^k)$ with respect to N, under a budget constraint $M(N+P) = B$, where g_i is the point-wise loss of our learning task at time-step i, e.g. $g_i(\theta, X_i,\dots,X_n,Y_i,\dots,Y_n) = (\xi_{i,n} - \varphi_\theta(X_i,Y_i))^2$ in our CVA case study, and φ_θ is the neural net (element of $\mathcal{NN}^+_{p+q,h,u,\varsigma}$) with parameters θ.

For ease of notation in this and the next part, we write $g_i(\theta, X^{k,l}, Y^k)$ and $g_i(\theta, X, Y)$ instead of $g_i(\theta, X_i^{k,l},...,X_n^{k,l},Y_i^k,...,Y_n^k)$ and $g_i(\theta, X_i,...,X_n,Y_i,...,Y_n)$ (it is then implied that X and Y play formally the role of vectors containing their path from time-step i up to n).

Proposition 2.4. *The hierarchical simulation factor that minimizes the variance of the loss* $\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} g_i(\theta, X^{k,l}, Y^k)$ *with respect to* N*, subject to the budget constraint* $M(N+P) = B$*, is*

$$N_i^\theta = \sqrt{\frac{Q_i^\theta P}{R_i^\theta}}, \tag{2.10}$$

where

$$\begin{aligned} R_i^\theta &= \mathbb{C}\mathrm{ov}(g_i(\theta, X^{1,1}, Y^1), g_i(\theta, X^{1,2}, Y^1)) = \mathbb{V}\mathrm{ar}(\mathbb{E}(g_i(\theta, X^{1,1}, Y^1)|Y^1)) \\ Q_i^\theta &= \mathbb{E}(\mathbb{V}\mathrm{ar}(g_i(\theta, X^{1,1}, Y^1)|Y^1)) = \mathbb{V}\mathrm{ar}(g_i(\theta, X^{1,1}, Y^1)) - R_i^\theta. \end{aligned}$$

Proof. After rearranging terms, one can show that

$$\mathbb{V}\mathrm{ar}\Big(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N} g_i(\theta, X^{k,l}, Y^k)\Big) = \frac{R_i^\theta}{B}\Big(\frac{1}{N}\big(N - \sqrt{\frac{Q_i^\theta P}{R_i^\theta}}\big)^2 + \big(\sqrt{\frac{Q_i^\theta}{R_i^\theta}} + \sqrt{P}\big)^2\Big),$$

where

$$\begin{aligned} Q_i^\theta &= \mathbb{E}[(g_i(\theta, X^{1,1}, Y^1))^2] - \mathbb{E}\left[g_i(\theta, X^{1,1}, Y^1)\, g_i(\theta, X^{1,2}, Y^1)\right] \\ R_i^\theta &= \mathbb{E}\left[g_i(\theta, X^{1,1}, Y^1)\, g(\theta, X^{1,2}, Y^1)\right] - (\mathbb{E}[g_i(\theta, X^{1,1}, Y^1)])^2. \end{aligned}$$

□

The ratio

$$\frac{Q_i^\theta}{R_i^\theta} = \frac{\mathbb{E}(\mathbb{V}\mathrm{ar}(g_i(\theta, X^{1,1}, Y^1)|Y^1))}{\mathbb{V}\mathrm{ar}(\mathbb{E}(g_i(\theta, X^{1,1}, Y^1)|Y^1))}$$

in (2.10) measures the relative contributions of Y and X to the variance of the loss estimator (note that $Q_i^\theta + R_i^\theta = \mathbb{V}\mathrm{ar}(g_i(\theta, X^{1,1}, Y^1))$, by the total variance formula). To estimate the values of Q_i^θ and of R_i^θ therein, one only needs to simulate $(X^{1,1}, X^{1,2}, Y^1)$, i.e., with respect to the bare simulation of (X, Y), one extra simulation of X conditional on each realization of Y.

As a fixed value of N has to be chosen throughout all the simulation and training task, for the above result to be of practical use, N_i^θ has to be reasonably stable with respect to both pricing time steps i and SGD iterations (the transfer learning scheme of Section 2.2.3 is advantageous in this respect in that it stabilizes the learning). If so, it leads to the following:

Heuristic 2.5. Choose for N the average of the values N_i^θ obtained during the SGD iterations and the time steps. Make for M the corresponding choice deduced from the budget constraint, i.e. $M = \frac{B}{N+P}$.

Note that N depends only on P, and M on P and Ψ. If P is not analytically known, it can be deduced from simulation times of experiments corresponding to the same M but different N. Namely, let B and B' the budgets corresponding to configurations (M, N) and (M, N'). We have

$$\frac{B}{B'} = \frac{P+N}{P+N'} \tag{2.11}$$

One can deduce P by identifying the ratio in (2.11) to that of the execution times of (M, N) and (M, N'). For doing so, it is preferable to choose M large enough to avoid time measurement noise that may be due to caching or parallelization of the simulations.

2.3.4 Statistical Convergence Analysis

In this part we completely omit the index i from the notation. For every possible parameterization $\theta \in \Theta \subset \mathbb{R}^d$ of our neural network (with d parameters), define:

$$\begin{aligned} G(\theta) &:= \mathbb{E}[g(\theta, X, Y)] \\ \hat{G}_{M,N}(\theta) &:= \frac{1}{MN} \sum_{k=1}^{M} \sum_{l=1}^{N} g(\theta, X^{k,l}, Y^k) \end{aligned}$$

and for all $\epsilon > 0$ and non-empty subsets E of Θ:

$$\begin{aligned} S^\epsilon(E) &:= \{\theta \in E : G(\theta) \leq \min_E G + \epsilon\} \\ \hat{S}^\epsilon_{M,N}(E) &:= \{\theta \in E : \hat{G}_{M,N}(\theta) \leq \min_E \hat{G}_{M,N} + \epsilon\} \end{aligned}$$

Let $\mathcal{X}$ and $\mathcal{Y}$ be the state spaces of X and Y. For all $\theta, \theta' \in \Theta$ and $t \in \mathbb{R}$, $y \in \mathcal{Y}$, denote:

$$\begin{aligned} \Gamma(\theta, \theta', X, Y) &:= g(\theta', X, Y) - g(\theta, X, Y) \\ \mathcal{M}(\theta, \theta', t, y) &:= \mathbb{E}\left[\exp\left(t\,\Gamma(\theta, \theta', X, Y)\right) | Y = y\right]. \end{aligned}$$

We are interested in the event

$$\{\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)\} = \bigcup_{\theta \in E \setminus S^{\epsilon}(E)} \bigcap_{\theta' \in E} \{\hat{G}_{M,N}(\theta) \leq \hat{G}_{M,N}(\theta') + \delta\}, \tag{2.12}$$

where E is a non-empty subset of Θ and $\epsilon, \delta > 0$. This is the event that a minimum of the finite-sample problem is far from being a minimum of the population mean minimization problem. Theorem 2.6 provides a bound on the probability of this event when E is finite.

Theorem 2.6. *Let E be a finite and non-empty subset of Θ and let $0 < \delta < \epsilon$. Assume that $E \setminus S^{\epsilon}(E) \neq \emptyset$ and that there exist $b_1, b_2 > 0$ such that for every $t \in \mathbb{R}$ and $\theta, \theta' \in E$:*

$$\mathcal{M}(\theta, \theta', t, Y) \leq \exp\left(\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y]\, t + \frac{b_1^2 t^2}{2}\right) \quad a.s. \tag{2.13}$$

$$\mathbb{E}[\exp\left(t\,\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y]\right)] \leq \exp\left(\mathbb{E}[\Gamma(\theta, \theta', X, Y)]\, t + \frac{b_2^2 t^2}{2}\right). \tag{2.14}$$

Then we have:

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)) < |E| \exp\left(\frac{-M(\epsilon - \delta)^2}{2\,(b_1^2/N + b_2^2)}\right).$$

Proof. See Section 2.5.1. □

Theorem 2.7 yields a similar bound valid for a possibly infinite parameter space, under additional assumptions of compactness and convexity of this space and Lipschitz continuity of the point-wise loss function. For brevity we write $S^{\epsilon}(\Theta) := S^{\epsilon}$, $\hat{S}^{\delta}_{M,N}(\Theta) := \hat{S}^{\delta}_{M,N}$.

Theorem 2.7. *Assume that Θ is a compact and convex and let $0 < \delta < \epsilon$. Let $D := \sup_{\theta, \theta' \in \Theta} \|\theta - \theta'\|$ and assume that there exists a mapping $L \colon \mathcal{X} \times \mathcal{Y} \to \mathbb{R}^{\star}_{+}$ such that $\mathbb{E}[\exp\left(t\,L(X, Y)\right)] < \infty$ for all t in some neighbourhood of 0 and for all $\theta, \theta' \in \Theta$:*

$$|g(\theta, X, Y) - g(\theta', X, Y)| \leq L(X, Y)\, \|\theta - \theta'\| \quad a.s. \tag{2.15}$$

Let $\bar{L} := \mathbb{E}[L(X, Y)]$ and assume that there exist $\ell_1, \ell_2 > 0$ such that:

$$|L(X, Y) - \mathbb{E}[L(X, Y)|Y]| \leq \ell_1 \quad a.s. \tag{2.16}$$

$$|\mathbb{E}[L(X, Y)|Y] - \bar{L}| \leq \ell_2 \quad a.s. \tag{2.17}$$

and that there exist $b_1, b_2 > 0$ such that for every $t \in \mathbb{R}$ and $\theta, \theta' \in \Theta$:

$$\mathcal{M}(\theta, \theta', t, Y) \leq \exp\left(\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y]\, t + \frac{b_1^2 t^2}{2}\right) \quad a.s.$$

$$\mathbb{E}[\exp\left(t\,\mathbb{E}[\Gamma(\theta, \theta', X, Y)|Y]\right)] \leq \exp\left(\mathbb{E}[\Gamma(\theta, \theta', X, Y)]\, t + \frac{b_2^2 t^2}{2}\right).$$

Then

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}) \leq \inf_{L' > \bar{L}} \left\{ \left(\left(\frac{8\,L'\,D}{\epsilon - \delta} + 1\right)^d + 1 \right) \exp\left(\frac{-M(\epsilon - \delta)^2}{8\,(b_1^2/N + b_2^2)}\right) + \exp\left(\frac{-M(L' - \bar{L})^2}{2\,(\ell_1^2/N + \ell_2^2)}\right) \right\}.$$

Proof. See Section 2.5.2. □

The Lipschitz assumptions of Theorem 2.7 are reasonable in our case since our neural network is Lipschitz with respect to its parameters, and its composition with the loss function remains Lipschitz if we assume that the parameters are bounded. In particular, these Lipschitz assumptions are satisfied in our learnings if we assume that *(i)* the processes X and Y are bounded (natively or after numerical truncation), *(ii)* the payoff function f (cf. (2.2)), which is embedded in the loss function g, is Lipschitz continuous or bounded, and *(iii)* Lipschitz continuous activation functions are used in the neural networks.

The following result can help in selecting M for reaching a target confidence level $(1-\alpha)$.

Corollary 2.8. *Let $0<\alpha<1$ and assume the conditions in Theorem 2.7 hold. Then choosing an arbitrary $L'>\bar{L}$, $0<u<1$, and*

$$M=\max\{\frac{8\,(b_1^2/N+b_2^2)}{(\epsilon-\delta)^2}\log(\frac{1}{u\,\alpha}\left((\frac{8\,L'\,D}{\epsilon-\delta}+1)^d+1\right)),\frac{8\,(\ell_1^2/N+\ell_2^2)}{(L'-\bar{L})^2}\log(\frac{1}{(1-u)\,\alpha})\}$$

we obtain that $\hat{S}^{\delta}_{M,N}\subset S^{\epsilon}$ with probability at least $(1-\alpha)$.

Proof. Choose any $L'>\bar{L}$ and $0<u<1$. Then it suffices that:

$$\begin{cases}\left((\frac{8\,L'\,D}{\epsilon-\delta}+1)^d+1\right)\exp(\frac{-M(\epsilon-\delta)^2}{8\,(\frac{b_1^2}{N}+b_2^2)}) & < \quad u\,\alpha\\ \exp(\frac{-M(L'-\bar{L})^2}{2\,(\frac{\ell_1^2}{N}+\ell_2^2)}) & < (1-u)\,\alpha\end{cases}$$

which is verified by choosing M as stated. □

As the above formula for M is decreasing in N, no matter how large N is, M has to be greater than the limit as $N\to\infty$, which provides a lower bound for M. This is natural as we do not expect to get an efficient sampling and good generalization just by increasing the number of realizations of X only. Accordingly:

Heuristic 2.9. In order to satisfy a constraint $\mathbb{Q}(\hat{S}^{\delta}_{M,N}\subset S^{\epsilon})\geq 1-\alpha$ instead of a target budget B, choose N as in Heuristic 2.5 (since it is independent of the budget value), then deduce M from Corollary 2.8.

In the data augmentation logic recalled before Remark 2.3, one could then set the size M of the mark-to-market data Y as a function of the hierarchical simulation factor N and of the confidence level $(1-\alpha)$. In a context where collecting the data Y is expensive, Corollary 2.8 would thus allow the user to benefit from the augmentation factor N through a reduction of the size M of the dataset for Y. However, making Heuristic 2.9 really practical would require to know (or estimate) the parameters b_1, b_2, ℓ_1, ℓ_2 and $\bar{L}$.

2.4 CVA Case Study

We now introduce a CVA case study, to be pursued in Section 2.4.4. In this context the reference probability measure represents a risk-neutral measure chosen by the market, to which the model is calibrated in mark-to-market terms.

2.4.1 Market and Credit Model

We consider a bank trading derivative contracts in different economies e with various clients c. The currency corresponding to the economy labeled by 0 is taken as the reference currency. Let there be given the short rate process $r^{\langle e\rangle}$ in each economy e, as well as the exchange rate process $\chi^{\langle e\rangle}$ from the currency of each economy $e \neq 0$ to the reference currency. Each client c of the bank has a stochastic default intensity process $\gamma^{\langle c\rangle}$ and a default-time $\tau^{\langle c\rangle}$. For notational convenience we also define $\chi^{\langle 0\rangle} = 1$ and we denote by $\gamma^{\langle 0\rangle}$ the default intensity of the bank itself. We consider an Euler-Marayama time-discretization of the model in Section 2.6. We use the same notation for the continuous-time processes and their discrete-time approximations (with time-step equal to 1year to alleviate the notation).

Remark 2.10. In practice, the time-discretizations are stepping through a refined simulation time grid. This simulation grid is also used when integrating numerically some of the above diffusions, e.g. the default intensities in (2.39), or for defining risk-neutral discount factors β_i associated with the reference currency by approximating $(-\ln \beta_i)$ using numerical integration[2.5] of $r^{\langle 0\rangle}$ on $[0, i]$. Learning, pricing and checking for default events, however, are only done at the coarser pricing time steps. Hence, although we step through the fine time grid in our discretized diffusions, we only need to store the values of the processes at the pricing time steps.

We define X as the collection of all the default indicator processes of the clients c and Y as the collection of all the short interest rate, FX, and default intensity processes r, χ and γ (except for the instrumental $\chi^{\langle 0\rangle} = 1$), endowed with the filtration generated by the innovation in the model, i.e. the collection of all the Gaussian and exponential variables involved at the increasing time steps $i \in \{1, \ldots, n\}$. Note that both Y (by itself) and (X, Y) (jointly) are Markov processes with respect to this filtration.

2.4.2 Learning the CVA

We denote by $\mathrm{MtM}_i^{\langle c\rangle}$ the mark-to-market at time i, from the point of view of the bank and in units of the reference currency, of all the contracts with the client c. By mark-to-market we mean trade additive counterparty-risk-free valuation, i.e. the risk-neutral conditional expectation of the future contractually promised cash flows, expressed in units of the reference currency and discounted at the risk-free rate $r^{\langle 0\rangle}$. We restrict ourselves to interest-rate derivatives for which mark-to-market valuation at i is a function of Y_i, by the nature of the cash-flows and the Markov property of Y[2.6]. The CVA of the bank then corresponds to the risk-neutral conditional expectation of its future risk-free discounted client default losses. Namely, the CVA of the bank at the time step i is given by[2.7]

$$\mathrm{CVA}_i = \sum_c \mathrm{CVA}_i^{\langle c\rangle} \mathbb{1}_{\{i < \tau^{\langle c\rangle}\}}, \tag{2.18}$$

2.5. It is also possible to jointly simulate exactly $r^{\langle 0\rangle}$ and its integral without the need for numerical integration, see for example Glasserman, 2004.

2.6. see Remark 2.13.

2.7. Assuming that the netting set for a given client is the whole set of transactions with this client.

for a (pre-default) CVA of the client c such that

$$\mathrm{CVA}_i^{\langle c\rangle} = \mathbb{E}\left[\sum_{j=i}^{n} \beta_i^{-1}\,\beta_{j+1}(\mathrm{MtM}_{j+1}^{\langle c\rangle})^+\,\mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1}\middle| X_i, Y_i\right]. \tag{2.19}$$

Hence $\mathrm{CVA}_i^{\langle c\rangle} = \mathbb{1}_{\{i<\tau^{\langle c\rangle}\}}\,\varphi_i^{\langle c\rangle}(Y_i)$, where (cf. (2.4) and (2.7))

$$\varphi_i^{\langle c\rangle} \in \mathrm{argmin}_{\varphi\in\mathcal{B}(\mathbb{R}^q)}\mathbb{E}\left[\left(\sum_{j=i}^{n-1} \beta_i^{-1}\,\beta_j(\mathrm{MtM}_j^{\langle c\rangle})^+\,\mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1} - \varphi(Y_i)\right)^2\middle| i<\tau^{(c)}\right]. \tag{2.20}$$

We also mention the following default intensity-based formula for the CVA of the client c (cf. [Albanese et al., 2021]):

$$\widetilde{\mathrm{CVA}}_i^{\langle c\rangle} = \mathbb{E}\left[\sum_{j=i}^{n-1} \beta_i^{-1}\,\beta_j(\mathrm{MtM}_j^{\langle c\rangle})^+\,\gamma_j^{\langle c\rangle}\exp\left(-\sum_{s=i}^{j-1}\gamma_s^{\langle c\rangle}\right)\middle| Y_i\right]\mathbb{1}_{\{i<\tau^{(c)}\}}, \tag{2.21}$$

which converges to the same continuous-time limit as $\mathrm{CVA}_i^{\langle c\rangle}$ when the time discretisation step[2.8] goes to zero. Hence $\widetilde{\mathrm{CVA}}_i^{\langle c\rangle} = \mathbb{1}_{\{i<\tau^{\langle c\rangle}\}}\,\tilde{\varphi}^{\langle c\rangle}(Y_i)$, where (cf. (2.4) and (2.7))

$$\tilde{\varphi}_i^{\langle c\rangle} \in \mathrm{argmin}_{\varphi\in\mathcal{B}(\mathbb{R}^q)}\mathbb{E}\left(\sum_{j=i}^{n-1} \beta_i^{-1}\,\beta_j(\mathrm{MtM}_i^{\langle c\rangle})^+\,\gamma_j^{\langle c\rangle}\exp\left(-\sum_{s=i}^{j-1}\gamma_s^{\langle c\rangle}\right) - \varphi(Y_i)\right)^2. \tag{2.22}$$

We reiterate that Algorithm 2.2 with hierarchical simulation of (X, Y) is generically applicable to all the XVA metrics. The focus on the CVA in our case study is for benchmarking purposes only. Were it for the CVA only, the regression learning scheme with minimal variance would obviously be the one based on (2.21), where a CVA is computed separately for each client based on its default intensity. At the other extreme of the spectrum, equivalently to (2.18)-(2.19), one can rewrite the CVA of the bank using a single expectation conditional on the default states of all clients, as

$$\mathrm{CVA}_i = \mathbb{E}\left[\sum_{c}\sum_{j=i}^{n} \beta_i^{-1}\,\beta_{j+1}(\mathrm{MtM}_{j+1}^{\langle c\rangle})^+\,\mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1}\middle| X_i, Y_i\right]. \tag{2.23}$$

Hence $\mathrm{CVA}_i = \varphi_i^\star(X_i, Y_i)$, where

$$\varphi_i^\star \in \mathrm{argmin}_{\varphi\in\mathcal{B}(\mathbb{R}^p\times\mathbb{R}^q)}\mathbb{E}\left[\left(\sum_{c}\sum_{j=i}^{n} \beta_i^{-1}\,\beta_{j+1}(\mathrm{MtM}_{j+1}^{\langle c\rangle})^+\,\mathbb{1}_{j<\tau^{\langle c\rangle}\le j+1} - \varphi(X_i, Y_i)\right)^2\right]. \tag{2.24}$$

On top of the learning schemes (2.22) and (2.24) associated with the formulations (2.21) and (2.23), another computational alternative in each case is nested Monte Carlo as detailed in [Abbas-Turki et al., 2018]. This variety of approaches will be useful for benchmarking purposes.

2.8. Conventionally set to one in this paper.

2.4.3 Preliminary Learning Results Based on IID Data

In the following experiments, we assume that a bank is trading derivatives in 10 economies with 8 clients. Implementing the discretized mark-to-market and credit model, we get a total of 10 interest rates, 9 cross-currency rates, and 8 default intensities. This yields 27 diffusive mark-to-market risk factors and 8 default indicator processes. For time-stepping, we use $n = 100$ pricing time steps and 25 simulation sub-steps per pricing time step (see Remark 2.10). We consider a portfolio of 500 interest rate swaps with random characteristics (notional, currency and counterparty), the $\mathrm{MtM}^{\langle c \rangle}$ are thus analytic. All swaps are priced at par at inception. For all the runs of the simulations in this section, whether they be for training or testing, we use $M = 16384$ paths for the market risk factors Y.

We implemented the learning procedure of Algorithm 2.2 in PyTorch with custom CUDA kernels for label generation during the backward iterations, the way detailed in Section 2.2.6 (and in the accompanying github repository). Moreover we implemented an optimized CUDA benchmark involving nested simulations, using the intensity-based formulation (2.21) for the inner CVA computations. For the nested Monte Carlo, with a view on the square-root rule recalled in [Abbas-Turki et al., 2018](Section 3.3), we used 128 inner paths. The nested Monte Carlo CVA is only computed at few pricing times due to the heavy calculation.

The comparison between the two panels of Figure 2.4 reveals a difficulty with the neural net learning approach of Algorithm 2.2 applied to the defaults-based formulation (2.23) on the basis of i.i.d. simulated data. In this case, represented by the left panel in Figure 2.4, the network only learns a rather crude and noisy approximation of the CVA conditional to each training time: it is only on the mean that the learned CVA agrees with the nested Monte Carlo estimator; on the tails it largely fails. As visible from the right panel, the CVA learned using the intensity-based formulation, instead, yields satisfactory results on a wide range of quantiles of the targeted distribution.

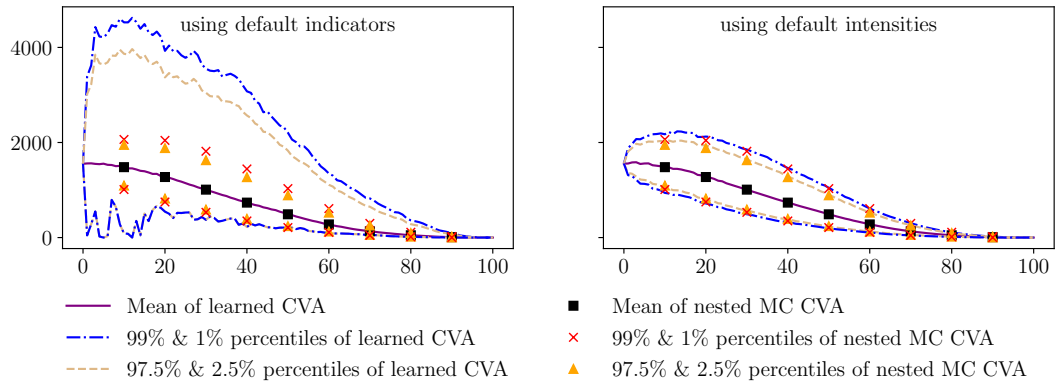

Figure 2.4. CVA learned using default indicators versus using default intensities (**X axis:** pricing times, **Y axis:** CVA levels). Statistics computed using out-of-sample paths.

2.4.4 Learning Results Based on Hierarchically Simulated Data

In order to improve the learning (2.24) of the defaults-based CVA (2.23) (cf. Section 2.4.3), we apply to it the hierarchical simulation technique of Section 2.3. Let $(r^1, \chi^1, \gamma^1), \ldots, (r^M, \chi^M, \gamma^M)$, be i.i.d sample paths of the triple of processes (r, χ, γ). Let $\{\epsilon^{k,l}, 1 \le k \le M, 1 \le l \le N\}$ be i.i.d samples of ϵ, the vector defined by the right-hand side in (2.39) where c ranges over clients. Then we can define MN samples of the vector of the default indicator processes of the clients at every pricing time i based on (2.39). Figure 2.5 illustrates the ensuing simulation scheme for the default indicator of a generic client of the bank, with sampled default times $\tau^{k,l}$.

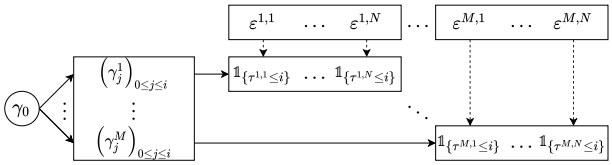

Figure 2.5. Default simulation scheme.

We then learn the CVA process at different time steps for the whole portfolio at once based on (2.23), trying different combinations of the number of market paths M and of the hierarchical simulation factor N. Figures 2.6 and 2.8 and show the relative RMSE of the trained neural network against the nested Monte Carlo benchmark[2.9], the simulation and training times on the GPU and the host RAM usage, as functions of the number of diffusion paths M and of the hierarchical simulation factor N. An estimation of the RMSE against the groundtruth CVA, without computing the latter, is also provided in Figure 2.7. We already see some configurations $(\frac{1}{2}M, N)$ being better than $(M, \frac{1}{2}N)$, as they achieve a similar accuracy with less memory footprint. For example, $(32768, 1024)$ is better than $(65536, 512)$, given that the former is 30% faster to simulate and price, while also occupying 23% less CPU memory. For the execution times in Figure 2.8, the runs were done on a server with an Intel Xeon Gold 6248 CPU and 4 Nvidia Tesla V100 GPUs (out of which we used only one). For performance comparison reasons, we use for all (M, N) configurations the same number of epochs $E = 8$ and number of batches $|\mathcal{B}| = 32$, which yields a total of 256 stochastic gradient descent steps during any training task.

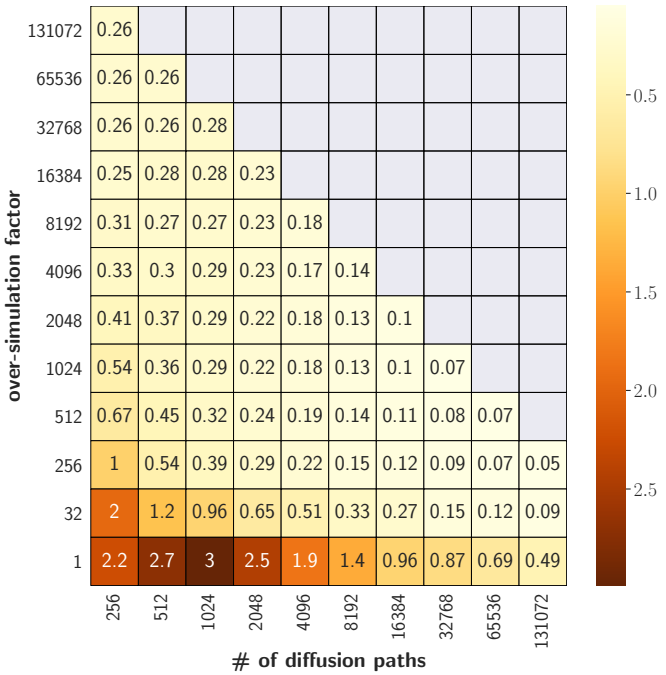

Figure 2.6. Relative RMSE of the prediction against a nested Monte Carlo benchmark at the pricing time $i = 5$ years, for different combinations of the number of market paths M and of the hierarchical simulation (or simply *over-simulation* in the figure) factor N, when the nested Monte Carlo benchmark is non-zero. The error here is a Monte Carlo estimate of $\sqrt{\mathbb{E}\left[\left(\frac{\mathrm{CVA}_{\mathrm{pred}} - \mathrm{CVA}_{\mathrm{nested}}}{\mathrm{CVA}_{\mathrm{nested}}}\right)^2\right]}$ where $\mathrm{CVA}_{\mathrm{pred}}$ is the CVA estimate predicted by the considered neural network given a state of market and default factors and $\mathrm{CVA}_{\mathrm{nested}}$ is a nested Monte Carlo estimator given the same state.

2.9. RMSE restricted to the realizations where the benchmark is non-zero.

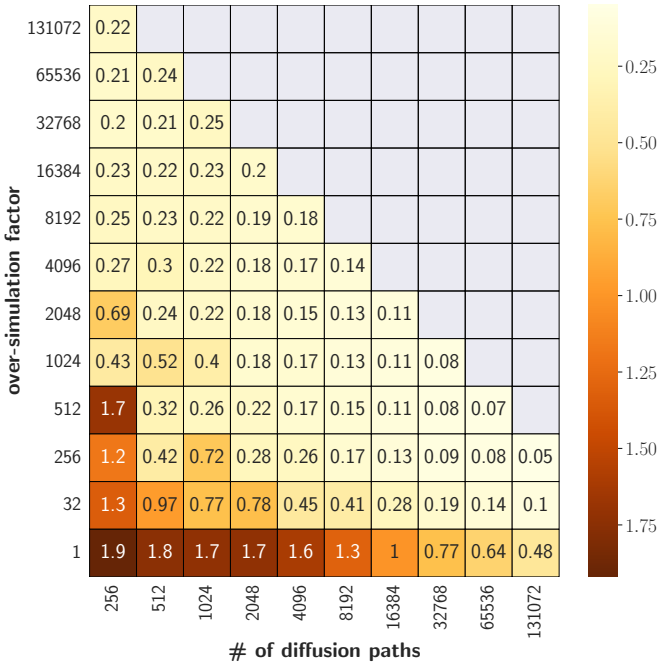

Figure 2.7. Relative RMSE of the prediction against the ground-truth CVA at the pricing time $i = 5$ years, computed by the twin simulation approach of Section 2.2.5 for different combinations of the number of market paths M and of the hierarchical simulation (or simply *over-simulation* in the figure) factor N. The error here is a Monte Carlo estimate of $\sqrt{\frac{\mathbb{E}[(\mathrm{CVA}_{\mathrm{pred}} - \mathrm{CVA}_{\mathrm{exact}})^2]}{\mathbb{E}[\mathrm{CVA}^2_{\mathrm{exact}}]}}$, where $\mathrm{CVA}_{\mathrm{pred}}$ is the CVA estimate predicted by the considered neural network given a state of market and default factors and $\mathrm{CVA}_{\mathrm{exact}}$ is the exact CVA (the value of which does not need to be computed, by virtue of (2.8)-(2.9)), given the same state.

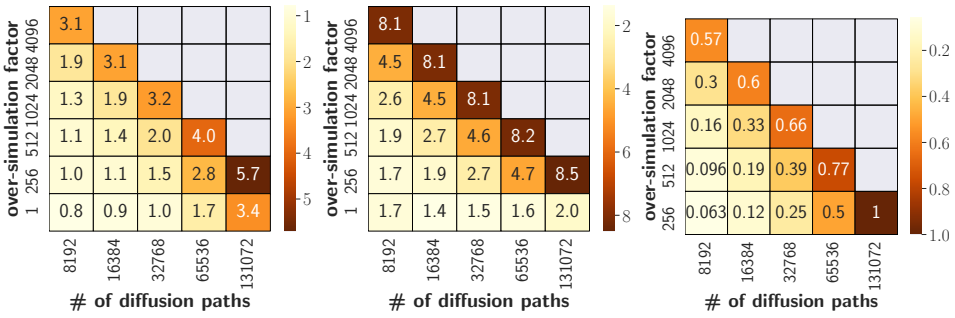

Figure 2.8. Simulation times in seconds **(left)** and training times in minutes **(center)** and RAM usage as a % of its maximum usage over all the displayed experiments **(right)**, for different combinations of the number of market paths M and of the hierarchical simulation factor N.

The dominance of the impact of the variance of X on that of ξ has been demonstrated

in Figure 2.3. Figure 2.9 shows the $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$ (cf. (2.10)) obtained in the base case $N=1$.

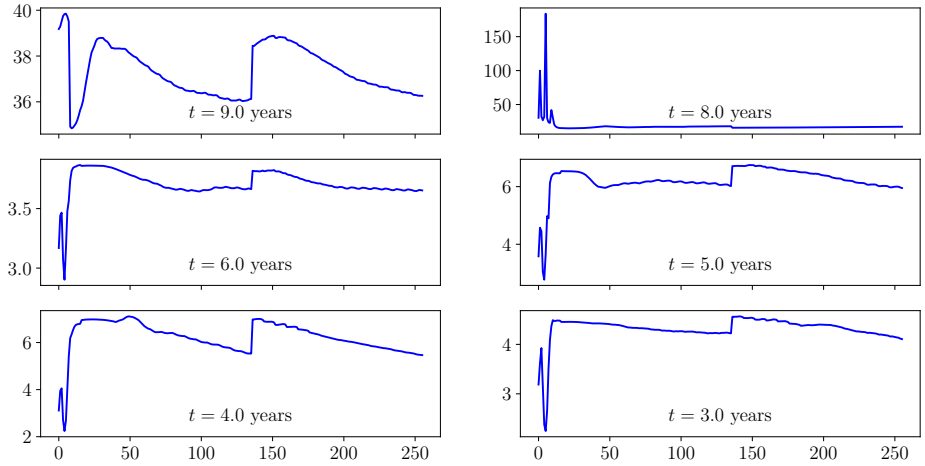

Figure 2.9. $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$ at different pricing time steps i **(panels)** and SGD iterations **(x axes)**.

With respect to the discussion introducing Heuristic 2.5, one can note that these are quite stable, of the order of a few tens, with respect to both pricing time steps i and SGD iterations. To obtain from the $\sqrt{\frac{Q_i^\theta}{R_i^\theta}}$ the N_i^θ in (2.10), one needs to multiply them by $\sqrt{P}$ (e.g. if a market simulation is 100 times slower than an ensuing default simulation, then the factors displayed in Figure 2.9 must be multiplied by 10). Solving the equation (2.11) for P on the basis of the columns $M=65536$ in Figures 2.6-2.8 yields $P\approx 497$. So the numbers in Figure 2.9 need to be multiplied by $\sqrt{497}\approx 22.3$ to get the optimal N as per Heuristic 2.5. In view of this, we expect an optimal hierarchical simulation factor N of the order of a few hundreds.

More results for $N=1, 32, 64, 128, 512$ are shown in Figure 2.10, which are to be compared to the right plot in Figure 2.4 obtained when learning the CVA relying on the intensity-based formula (2.21). In line with the above expectations, one needs $N=512$ in order to have a close enough match between the 1, 2.5, 97.5 and 99-th percentiles of the CVA learned from defaults and those of the nested Monte Carlo estimator (or of the intensity-based CVA learner represented by the right panel in Figure 2.4).

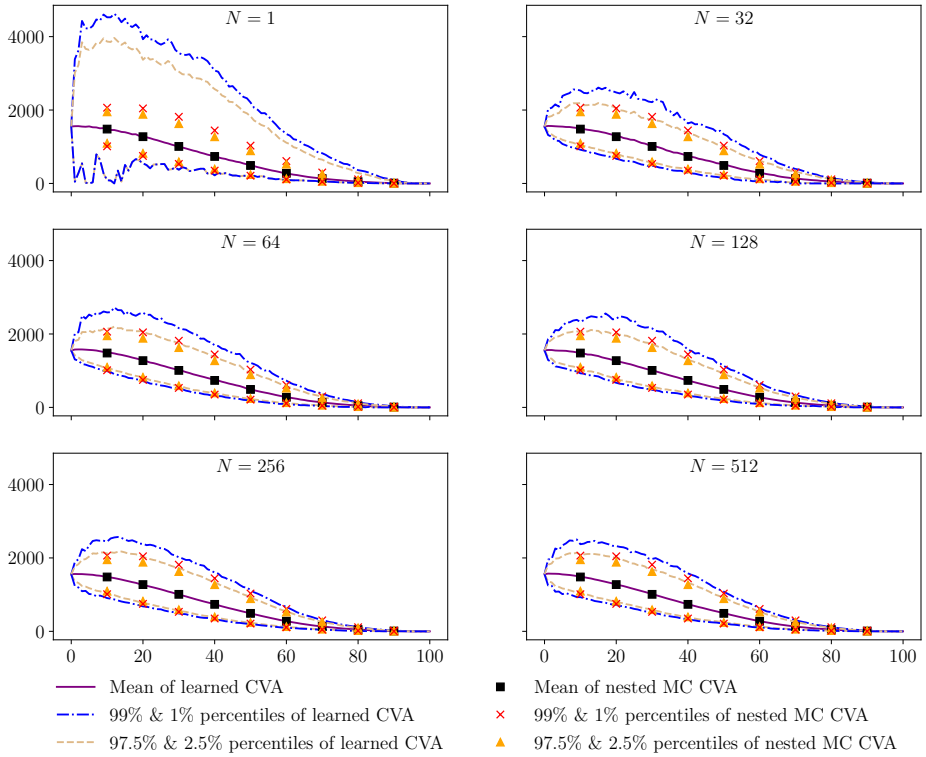

Figure 2.10. Learned and nested Monte Carlo CVA processes for various hierarchical simulation factors N (***x*-axis:** pricing times, ***y*-axis:** CVA levels; $M = 16384$). Statistics computed using out-of-sample paths.

These results show that hierarchical simulation is essential to a defaults-based CVA learner.

Even after writing an optimized GPU implementation for the nested Monte Carlo estimator, it takes at least 32 minutes on the same hardware as above to compute that estimator for $M = 16384$ and $\sqrt{M} = 128$ inner paths[2.10], compared to approximately 8 minutes in the case of the learning approach with a very high hierarchical simulation factor ($N = 2048$). Moreover, going to higher XVA layers such as the FVA and the KVA, a nested Monte Carlo approach would become $\sqrt{M}$ times slower per each new layer [Abbas-Turki et al., 2018], whereas a regression approach would just become slower by a constant each time a new XVA layer is added. In addition, learned XVA metrics can be used in predictionat a very low cost (inference is very fast as it involves no automatic differentation or stochastic gradient descent), whereas nested Monte Carlo numbers must be recomputed from scratch every time.

2.4.5 Conclusion

Having in mind a portfolio of the order of one million trades spread over maturities ranging over 50 years and involving a few thousands of clients, the computational CPU ressources typically available in banks hardly allow considering a mark-to-market cube with more than 10^4 paths. Switching to GPU ressources (as required anyway if training path-wise

2.10. However, when doing the error computations and in all plots, we used 1024 inner paths to get benchmark CVAs that are sufficiently accurate *point-wise* and be able to get accurate tail estimates, and nested Monte Carlo simulation thus takes 8 times more computation time.

XVA metrics is envisioned) could allow computing a mark-to-market cube with 10^5 to 10^6 paths (in about one hour of computations spread over a few GPUs). Moreover, while we performed our computations using only one GPU, we expect a bank to have access to more than just a single GPU, which would drastically reduce the computation times given that Monte Carlo simulations and stochastic gradient descent can easily be adapted to multi-GPU setups.

The bottom row of Figure 2.6 and the first plot ($N=1$) of Figure 2.10 illustrate that a path-wise CVA cannot be learned based on the hybrid market and defaults formulation (2.23): for $M=16384$ and 131072, the corresponding errors with respect to the banchmark nested Monte Carlo are 96% and 49%. However, increasing N from 1 (bottom row) to 256 brings these errors down to 11% and to 5%, while for $M=1024$ and $N=512$ the error is 1%. As visible from Figure 2.8, the simulation times are only marginally increased when increasing the hierarchical simulation factor N (while increasing the number of diffusion paths M increases the simulation time approximately by the same factor of increase in M). These results show that the hierarchical simulation technique is key to the success of a learning approach involving a combination of mark-to-market and default data.

2.5 Technical Proofs

The following proofs use arguments from Shapiro et al., 2014 and extend similar results to the conditionally independent, hierarchical simulation case. Theorem 2.6 extends the finite case in [Shapiro et al., 2021](Section 5.3.1). The major modifications in the proof are the use of a conditional moment generating function, the establishment of a large deviation upper-bound based on it, and the strict convexity of $\log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)$ with respect to t which becomes more technical in the conditional case. Then, similar to [Shapiro et al., 2021](Section 5.3.2), Theorem 2.7 extends these results to the infinite and bounded case, by Lipschitz continuity arguments. In both cases we rely on the following:

Lemma 2.11. *Let $\varphi\colon \mathcal{X}\times\mathcal{Y}\to R$ be such that $\varphi(X,Y)$ is integrable, does not degenerate to a constant and that, for all $z\in\mathbb{R}$ and $y\in V$, $\mathcal{M}(z,y):=\mathbb{E}\left[\exp\left(z\,\varphi(X,Y)\right)|Y=y\right]$ is well-defined. Then the Fenchel conjugate $I_N\colon a\mapsto\sup_{t\in\mathbb{R}}\left\{t\,a-\log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)\right\}$ of $t\mapsto\log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)$ is well-defined and*

$$\frac{1}{M}\log\left\{\mathbb{Q}\left(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\geq a\right)\right\}\leq -I_N(a)\quad \textit{for all } a>\mathbb{E}[\varphi(X,Y)] \tag{2.25}$$

where

$$I_N(a)=\frac{(a-\mathbb{E}[\varphi(X,Y)])^2}{\frac{1}{N}\mathbb{E}[\mathrm{Var}(\varphi(X,Y)|\xi)]+\mathrm{Var}(\mathbb{E}[\varphi(X,Y)|Y])}+o(|a-\mathbb{E}[\varphi(X,Y)]|^2) \tag{2.26}$$

Proof. Let $t>0$. Applying the Markov inequality, we have:

$$\begin{aligned}\mathbb{Q}\left(\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\geq a\right) &= \mathbb{Q}\left(\exp\left(\frac{t}{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\right)\geq\exp\left(Mta\right)\right)\\ &\leq \exp\left(-Mta\right)\mathbb{E}\left[\exp\left(\frac{t}{N}\sum_{k=1}^{M}\sum_{l=1}^{N}\varphi(X^{k,l},Y^k)\right)\right].\end{aligned} \tag{2.27}$$

For every $i \in \{1,\dots,M\}$, denote $Z_N^i := \exp\left(\frac{t}{N}\sum_{j=1}^N \varphi(X^{i,j}, Y^i)\right)$. By using the tower property repeatedly, one can show recursively that for all $i \in \{1,\dots,M\}$, denoting $\mathcal{Z}_i := \sigma(Z_{M,N}^1,\dots,Z_{M,N}^{M-i}, Y^{M-i+1},\dots Y^M)$:

$$\begin{aligned}\mathbb{E}[\exp\left(\frac{t}{N}\sum_{k=1}^M\sum_{l=1}^N \varphi(X^{k,l},Y^k)\right)] &= \mathbb{E}\left[\mathbb{E}\left[\left(\prod_{k=1}^{M-i+1} Z_N^k\right)\left(\prod_{k=1}^{i-1}\mathcal{M}(\frac{t}{N},Y^{M-k+1})^N\right)|\mathcal{Z}_i\right]\right]\\ &= \mathbb{E}\left[\left(\prod_{k=1}^{M-i} Z_N^k\right)\left(\prod_{k=1}^{i-1}\mathcal{M}\Big(\frac{t}{N},Y^{M-k+1}\Big)^N\right)\mathbb{E}[Z_N^{M-i+1}|Y^{M-i+1}]\right]\\ &= \mathbb{E}\left[\left(\prod_{k=1}^{M-i} Z_N^k\right)\left(\prod_{k=1}^{i}\mathcal{M}(\frac{t}{N},Y^{M-k+1})^N\right)\right].\end{aligned} \tag{2.28}$$

In particular, this identity for $i = M$ yields (recalling Y and the Y^k are i.i.d.)

$$\mathbb{E}[\exp\left(\frac{t}{N}\sum_{k=1}^M\sum_{l=1}^N \varphi(X^{k,l},Y^k)\right)] = (\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N])^M.$$

Hence, by (2.27),

$$\frac{1}{M}\log\{\mathbb{Q}(\frac{1}{MN}\sum_{k=1}^M\sum_{l=1}^N \varphi(X^{k,l},Y^k)\ge a)\} \le -t\,a + \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right).$$

The inequality being true for arbitrary $t>0$, taking the infimum over $t>0$ on the RHS yields

$$\frac{1}{M}\log\left(\mathbb{Q}(\frac{1}{MN}\sum_{k=1}^M\sum_{l=1}^N \varphi(X^{k,l},Y^k)\ge a)\right) \le -\sup_{t>0}\Big(t\,a - \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)\Big).$$

In order to establish (2.25), it remains to show that $t\mapsto \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)$ is convex and that $I_N(a) = \sup_{t>0}\{t\,a - \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)\}$.

Define $\Lambda(t) := \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N},Y)^N]\right)$. As a moment generating function is infinitely differentiable on its domain of definition, Λ is infinitely differentiable. After computations we get $\Lambda'(0) = \mathbb{E}[\varphi(X,Y)]$ and $\Lambda''(t) = \frac{\det(\mathbb{E}[A])}{\mathbb{E}[\mathbb{E}[V|Y]^N]^2}$ for the 2×2 random matrix

$$A = \left[\begin{array}{c|c} \frac{1}{N}\mathbb{E}\left[U^2V|Y\right]\mathbb{E}[V|Y]^{N-1} + (1-\frac{1}{N})\,\mathbb{E}\left[UV|Y\right]^2\mathbb{E}[V|Y]^{N-2} & \mathbb{E}\left[UV|Y\right]\mathbb{E}[V|Y]^{N-1} \\ \hline \mathbb{E}\left[UV|Y\right]\mathbb{E}[V|Y]^{N-1} & \mathbb{E}[V|Y]^N \end{array}\right],$$

where $U := \varphi(X,Y)$ and $V := \exp\left(\frac{t}{N}\,\varphi(X,Y)\right)$. We have:

$$A = \mathbb{E}[V|Y]^{N-2}\left[\begin{array}{c|c} \frac{1}{N}\mathbb{E}\left[U^2V|Y\right]\mathbb{E}[V|Y] + (1-\frac{1}{N})\,\mathbb{E}\left[UV|Y\right]^2 & \mathbb{E}\left[UV|Y\right]\mathbb{E}[V|Y] \\ \hline \mathbb{E}\left[UV|Y\right]\mathbb{E}[V|Y] & \mathbb{E}[V|Y]^2 \end{array}\right]$$

Let $\alpha,\beta\in\mathbb{R}$ and

$$\begin{aligned}\Delta_{\alpha,\beta} :=& \frac{1}{\mathbb{E}[V|Y]^{N-2}}\,[\alpha,\beta]\,A\begin{bmatrix}\alpha\\ \beta\end{bmatrix}\\ =& \alpha^2(\frac{1}{N}\mathbb{E}\left[U^2V|Y\right]\mathbb{E}[V|Y] + (1-\frac{1}{N})\,\mathbb{E}\left[UV|Y\right]^2) + \beta^2\mathbb{E}[V|Y]^2 + 2\,\alpha\,\beta\,\mathbb{E}\left[UV|Y\right]\mathbb{E}[V|Y].\end{aligned}$$

From the Cauchy-Schwarz inequality, we have:

$$\mathbb{E}\left[U^2 V|Y\right]\mathbb{E}[V|Y] \geq \mathbb{E}\left[UV|Y\right]^2. \tag{2.29}$$

We then have:

$$\begin{aligned}\Delta_{\alpha,\beta} &\geq \alpha^2\,\mathbb{E}\left[UV|Y\right]^2 + \beta^2\,\mathbb{E}[V|Y]^2 + 2\,\alpha\,\beta\,\mathbb{E}\left[UV|Y\right]\mathbb{E}[V|Y] \\ &= (\alpha\,\mathbb{E}\left[UV|Y\right] + \beta\mathbb{E}[V|Y])^2 \geq 0.\end{aligned}$$

Furthermore, we have $\Delta_{\alpha,\beta} > 0$ because $\Delta_{\alpha,\beta} = 0$ would imply equality in (2.29), which in turn is only attained when $U = 1$ *a.s.*, contradicting the non-degeneracy assumption made on φ. Therefore A is a.s. positive definite. Hence $\mathbb{E}[A]$ is positive definite, i.e. $\det(\mathbb{E}[A]) > 0$. In conclusion, Λ is strictly convex.

Let $\psi(t) := t\,a - \Lambda(t)$. For $a > \mathbb{E}[\varphi(X,Y)]$, we have

$$\psi'(0) = a - \Lambda'(0) = a - \mathbb{E}[\varphi(X,Y)] > 0.$$

Therefore there exists some $\varepsilon > 0$ such that $\psi'(t) > 0$ for all $t \in (0,\varepsilon)$. Hence, for all $t \in (0, \varepsilon)$, we have $t\,a - \Lambda(t) > 0$.

On the other hand, we have:

$$\sup_{t<0}\{t\,a - \Lambda(t)\} = \sup_{t<0}\{\underbrace{t\,(a - \mathbb{E}[\varphi(X,Y)])}_{<0} + t\,\mathbb{E}[\varphi(X,Y)] - \Lambda(t)\}.$$

Using convexity and concavity inequalities, we have

$$\begin{aligned}\Lambda(t) &= \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N]\right) \\ &\geq N\log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)]\right) \\ &\geq t\,\mathbb{E}[\varphi(X,Y)].\end{aligned}$$

We then obtain that

$$\sup_{t<0}\{t\,a - \Lambda(t)\} \leq 0.$$

Thus

$$\sup_{t>0}\{t\,a - \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N]\right)\} = \sup_{t\in\mathbb{R}}\{t\,a - \log\left(\mathbb{E}[\mathcal{M}(\frac{t}{N}, Y)^N]\right)\} = I_N(a),$$

which finishes to prove (2.25). As the Fenchel conjugate of the twice differentiable and strictly convex function Λ, I_N is twice differentiable and we have

$$\begin{aligned}I_N(\mathbb{E}[\varphi(X,Y)]) &= -\Lambda(0) = 0 \\ I_N'(\mathbb{E}[\varphi(X,Y)]) &= 0 \\ I_N''(\mathbb{E}[\varphi(X,Y)]) &= \frac{1}{\Lambda''(0)} = \frac{1}{\frac{1}{N}\mathbb{E}[\mathrm{Var}(\varphi(X,Y)|Y)] + \mathrm{Var}(\mathbb{E}[\varphi(X,Y)|Y])}.\end{aligned}$$

Hence a Taylor expansion around $\mathbb{E}[\varphi(X,Y)]$ gives (2.26). □

Remark 2.12. The Taylor approximation for the Fenchel conjugate in Lemma 2.11 would become exact if we added the hypothesis that $\varphi(X,Y)$ is Gaussian conditionally on Y and that $\mathbb{E}[\varphi(X,Y)|Y]$ is Gaussian. However, we can still obtain a similar expression as an exact lower-bound if we just assume sub-Gaussianity as in Theorem 2.7 (see the end of Section 2.5.1).

2.5.1 Proof of Theorem 2.6

Under the assumptions of Theorem 2.6, we have in view of (2.12):

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}(E) \not\subset S^{\epsilon}(E)) \leq \sum_{\theta\in E\setminus S^{\epsilon}(E)} \mathbb{Q}(\bigcap_{\theta'\in E}\{\hat{G}_{M,N}(\theta) \leq \hat{G}_{M,N}(\theta') + \delta\}) \tag{2.30}$$

Let us consider a minimizer $\theta^\star$ of G over E, hence

$$\forall \theta \in E \setminus S^\epsilon, G(\theta^\star) < G(\theta) - \epsilon. \tag{2.31}$$

We then have:

$$\mathbb{Q}(\hat{S}^\delta_{M,N}(E) \not\subset S^\epsilon(E)) \leq \sum_{\theta \in E \setminus S^\epsilon(E)} \mathbb{Q}(\hat{G}_{M,N}(\theta) \leq \hat{G}_{M,N}(\theta^\star) + \delta) \tag{2.32}$$

Define:

$$\hat{\Gamma}_{M,N}(\theta) := \frac{1}{M,N} \sum_{k=1}^{M} \sum_{l=1}^{N} \Gamma(\theta^\star, \theta, X^{k,l}, Y^k) = \hat{G}_{M,N}(\theta^\star) - \hat{G}_{M,N}(\theta)$$

Inequality (2.32) can then be rewritten:

$$\mathbb{Q}(\hat{S}^\delta_{M,N} \not\subset S^\epsilon(E)) \leq \sum_{\theta \in E \setminus S^\epsilon(E)} \mathbb{Q}(\hat{\Gamma}_{M,N}(\theta) \geq -\delta). \tag{2.33}$$

Now observe that

$$\mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)] = G(\theta^\star) - G(\theta) < -\epsilon < -\delta \tag{2.34}$$

for all $\theta \in E$. Hence, from inequality (2.33) and Lemma 2.11, we obtain

$$\begin{aligned} \mathbb{Q}(\hat{S}^\delta_{M,N}(E) \not\subset S^\epsilon(E)) &\leq |E| \max_{\theta \in E \setminus S^\epsilon(E)} \{\mathbb{Q}(\hat{\Gamma}_{M,N}(\theta) \geq -\delta)\} \\ &\leq |E| \exp\left(-M \min_{\theta \in E \setminus S^\epsilon(E)} \{I^\theta_N(-\delta)\}\right), \end{aligned} \tag{2.35}$$

where I^θ_N is the Fenchel conjugate of $\log\left(\mathbb{E}[\mathcal{M}^\theta(\frac{t}{N}, Y)^N]\right)$ and

$$\mathcal{M}^\theta(z, y) := \mathbb{E}\left[\exp\left(z\,\Gamma(\theta^\star, \theta, X, Y)\right) | Y = y\right]$$

for every $\theta \in E$. From inequalities (2.13) and (2.14), we get for every $\theta \in E \setminus S^\epsilon(E)$ and $t \in \mathbb{R}$:

$$\log\left(\mathbb{E}[\mathcal{M}^\theta(\frac{t}{N}, Y)^N]\right) \leq \mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)]\, t + \frac{1}{2}\left(\frac{b_1^2}{N} + b_2^2\right) t^2$$

Thus, for every $\theta \in E \setminus S^\epsilon(E)$, we have:

$$\begin{aligned} I^\theta_N(-\delta) &= \sup_{t \in \mathbb{R}} \{t\left(-\delta - \mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)]\right) + \mathbb{E}[\Gamma(\theta^\star, \theta, X, Y)]\, t - \log\left(\mathbb{E}[\mathcal{M}^\theta(\frac{t}{N}, Y)^N]\right)\} \\ &\geq \sup_{t \in \mathbb{R}} \{t\left(-\delta - \mathbb{E}[\Gamma(\theta^\star, \theta, \xi, \Gamma)]\right) - \frac{1}{2}\left(\frac{b_1^2}{N} + b_2^2\right) t^2\} \\ &= \frac{\left(-\delta - \mathbb{E}[\Gamma(\theta^\star, \theta, \xi, \Gamma)]\right)^2}{2\left(\frac{b_1^2}{N} + b_2^2\right)} \\ &> \frac{(\epsilon - \delta)^2}{2\left(\frac{b_1^2}{N} + b_2^2\right)} \end{aligned} \tag{2.36}$$

where the last inequality comes from (2.34). This yields the result.

2.5.2 Proof of Theorem 2.7

From the Lipschitz assumption (2.15), it follows that for all $\theta, \theta' \in \Theta$:

$$|\hat{G}_{M,N}(\theta) - \hat{G}_{M,N}(\theta')| \leq \hat{L}_{M,N} \|\theta - \theta'\| \quad \text{a.s.}$$

and

$$|G(\theta)-G(\theta')|\leq \mathbb{E}\left[L\left(X,Y\right)\right]\|\theta-\theta'\|$$

where

$$\hat{L}_{M,N}:=\frac{1}{MN}\sum_{k=1}^{M}\sum_{l=1}^{N}L\left(X^{k,l},Y^{k}\right).$$

Let $0<\delta'<\epsilon'$ and let $\Theta':=\{\theta_1,\ldots,\theta_C\}$ be a minimal ϱ-covering of Θ, for a given $\varrho>0$. We then have (cf. [Vershynin, 2018](Corollary 4.2.13 p.85)):

$$C\leq(\frac{2\,D}{\varrho}+1)^{d}.$$

Let $\tilde{\Theta}:=\Theta'\cup\{\theta^{\star}\}$, where $\theta^{\star}\in\mathrm{Argmin}_{\theta\in\Theta}G(\theta)$. We have $|\tilde{\Theta}|\leq(\frac{2\,D}{\varrho}+1)^{d}+1$. Theorem 2.6 yields:

$$\mathbb{Q}(\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})\not\subset S^{\epsilon'}(\tilde{\Theta}))<\left((\frac{2\,D}{\varrho}+1)^{d}+1\right)\exp\left(\frac{-M\,(\epsilon'-\delta')^{2}}{2\,(\frac{b_1^2}{N}+b_2^2)}\right).$$

Our next goal is to show the following assertion for suitable choices of δ' and ϵ' and for any $L'>\bar{L}$ (note that $\bar{L}<\infty$):

$$\{\hat{S}^{\delta}_{M,N}\not\subset S^{\epsilon}\}\cap\{L'\geq\hat{L}_{M,N}\}\subset\{\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})\not\subset S^{\epsilon'}(\tilde{\Theta})\}. \tag{2.37}$$

Let $L'>\bar{L}$ and assume that $\hat{S}^{\delta}_{M,N}\not\subset S^{\epsilon}$ and $L'\geq\hat{L}_{M,N}$, and that $\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})\subset S^{\epsilon'}(\tilde{\Theta})$. In particular, there exists $\theta\in\Theta$ such that $\hat{G}_{M,N}(\theta)\leq\min_{\Theta}\hat{G}_{M,N}+\delta$ and $G(\theta)>\min_{\Theta}G+\epsilon$. Let then $\theta'\subset\tilde{\Theta}$ such that $\|\theta-\theta'\|<\varrho$. We have:

$$\hat{G}_{M,N}(\theta')\leq\hat{G}_{M,N}(\theta)+L'\,\varrho\leq\min_{\Theta}\hat{G}_{M,N}+\delta+L'\,\varrho.$$

Thus, if we choose $\delta'=\delta+L'\,\varrho$, then we have $\hat{G}_{M,N}(\theta')\leq\min_{\Theta}\hat{G}_{M,N}+\delta'$, and consequently $\theta'\in\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})$ (as $\tilde{\Theta}\subset\Theta$). Hence, by our assumption $\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})\subset S^{\epsilon'}(\tilde{\Theta})$, we get that $\theta'\in S^{\epsilon'}(\tilde{\Theta})$. Thus:

$$G(\theta)\leq G(\theta')+L'\,\varrho\leq\min_{\tilde{\Theta}}G+\epsilon'+L'\,\varrho=\min_{\Theta}G+\epsilon'+L'\,\varrho,$$

as $\min_{\tilde{\Theta}}G=\min_{\Theta}G$ (since $\theta^{\star}\in\tilde{\Theta}$). Hence, if we also choose $\epsilon'=\epsilon-L'\,\varrho$ with ϱ such that $\varrho<\frac{\epsilon-\delta}{2\,L'}$ in order to ensure that $0<\delta'<\epsilon'$, then $G(\theta)\leq\min_{\Theta}G+\epsilon$, which contradicts our assumption and proves (2.37). Thus

$$\mathbb{Q}(\{\hat{S}^{\delta}_{M,N}\not\subset S^{\epsilon}\}\cap\{L'\geq\hat{L}_{M,N}\})\leq\mathbb{Q}(\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})\not\subset S^{\epsilon'}(\tilde{\Theta})).$$

As a consequence,

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}\not\subset S^{\epsilon})\leq\mathbb{Q}(\hat{S}^{\delta'}_{M,N}(\tilde{\Theta})\not\subset S^{\epsilon'}(\tilde{\Theta}))+\mathbb{Q}(\hat{L}_{M,N}>L').$$

But applying Hoeffding's lemma on the inequalities (2.16) and (2.17), Lemma 2.11, and proceeding similarly as in (2.36) to establish a lower bound for the Legendre transform, yields

$$\mathbb{Q}(\hat{L}_{M,N}>L')\leq\exp\left(\frac{-M\,(L'-\bar{L})^{2}}{2\,(\frac{\ell_1^2}{N}+\ell_2^2)}\right) \tag{2.38}$$

Thus,

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N}\not\subset S^{\epsilon})\leq\left((\frac{2\,D}{\varrho}+1)^{d}+1\right)\exp\left(\frac{-M\,(\epsilon'-\delta')^{2}}{2\,(\frac{b_1^2}{N}+b_2^2)}\right)+\exp\left(\frac{-M\,(L'-\bar{L})^{2}}{2\,(\frac{\ell_1^2}{N}+\ell_2^2)}\right).$$

We have $\epsilon' - \delta' = \epsilon - \delta - 2\,L'\,\varrho$. Finally, if we choose $\varrho = \frac{\epsilon-\delta}{4\,L'}$, then we get $\epsilon' - \delta' = \frac{\epsilon-\delta}{2}$ and

$$\mathbb{Q}(\hat{S}^{\delta}_{M,N} \not\subset S^{\epsilon}) \leq \left((\frac{8\,L'\,D}{\epsilon-\delta}+1)^d + 1 \right) \exp\left(\frac{-M\,(\epsilon-\delta)^2}{8\,(\frac{b_1^2}{N}+b_2^2)}\right) + \exp\left(\frac{-M\,(L'-\bar{L})^2}{2\,(\frac{\ell_1^2}{N}+\ell_2^2)}\right).$$

This concludes our proof.

2.6 Market and Credit Model in Continuous Time

For every economy e, the short-rate $r^{\langle e\rangle}$ and the exchange rate $\chi^{\langle e\rangle}$ against the reference currency respectively follow Vasicek and log-normal dynamics

$$\begin{aligned} \mathrm{d}r_t^{\langle e\rangle} &= a^{\langle e\rangle}\,(b^{\langle e\rangle} - r_t^{\langle e\rangle})\mathrm{d}t + \sigma^{r,\langle e\rangle}\mathrm{d}\tilde{B}_t^{r,\langle e\rangle} \\ \mathrm{d}\log\chi_t^{\langle e\rangle} &= (r_t^{\langle 0\rangle} - r_t^{\langle e\rangle} - \frac{1}{2}|\sigma^{\chi,\langle e\rangle}|^2)\mathrm{d}t + \sigma^{\chi,\langle e\rangle}\mathrm{d}B_t^{\chi,\langle e\rangle}. \end{aligned}$$

For both the bank ("$c=0$") and every counterparty $c(\neq 0)$, the process $\gamma^{\langle c\rangle}$ (funding spread for $c=0$ and default intensity for $c\geq 1$) follows CIR dynamics

$$\mathrm{d}\gamma_t^{\langle c\rangle} = \alpha^{\langle c\rangle}\,(\delta^{\langle c\rangle} - \gamma_t^{\langle c\rangle})\mathrm{d}t + \nu^{\langle c\rangle}\sqrt{\gamma_t^{\langle c\rangle}}\mathrm{d}B_t^{\gamma,\langle c\rangle}.$$

In the above, for every e, $\tilde{B}^{r,\langle e\rangle}$ is a $\mathbb{Q}^{\langle e\rangle}$ Brownian motion and, for every client c and economy e, $B^{\chi,\langle e\rangle}$ and $B^{\gamma,\langle c\rangle}$ are $\mathbb{Q}^{\langle 0\rangle}$ Brownian motions. Here $\mathbb{Q}^{\langle e\rangle}$ is the risk-neutral measure corresponding to the numeraire $e^{\int_0^{\cdot} r_s^{\langle e\rangle}\mathrm{d}s}$, and $a^{\langle\cdot\rangle}$, $b^{\langle\cdot\rangle}$, $\sigma^{\cdot,\langle\cdot\rangle}$, $\alpha^{\langle\cdot\rangle}$, $\delta^{\langle\cdot\rangle}$, $\nu^{\langle\cdot\rangle}$ are model parameters calibrated using liquid market instruments.

In line with the fundamental theorem of asset pricing, for any asset Z priced in a foreign currency $e\geq 1$, $\exp\left(-\int_0^{\cdot} r_s^{\langle e\rangle}\mathrm{d}s\right) Z$ and $^{-\int_0^{\cdot} r_s^{\langle 0\rangle}\mathrm{d}s}\chi^{\langle e\rangle}\,Z$ are martingales with respect to $\mathbb{Q}^{\langle e\rangle}$ and $\mathbb{Q}^{\langle 0\rangle}$ respectively. In particular,

$$\mathbb{E}^{\mathbb{Q}^{\langle e\rangle}}[e^{-\int_0^t r_s^{\langle e\rangle}\mathrm{d}s}\,Z_t] = Z_0 = \frac{1}{\chi_0^{\langle e\rangle}}\,\mathbb{E}^{\mathbb{Q}^{\langle 0\rangle}}[e^{-\int_0^t r_s^{\langle 0\rangle}\mathrm{d}s}\,\chi_t^{\langle e\rangle}\,Z_t].$$

Thus,

$$\frac{\mathrm{d}\mathbb{Q}^{\langle e\rangle}}{\mathrm{d}\mathbb{Q}^{\langle 0\rangle}}_{|t} = \exp\left(\int_0^t (r_s^{\langle e\rangle} - r_s^{\langle 0\rangle})\mathrm{d}s\right)\frac{\chi_t^{\langle e\rangle}}{\chi_0^{\langle e\rangle}} = \exp\left(-\frac{1}{2}(\sigma^{\chi,\langle e\rangle})^2\,t + \sigma^{\chi,\langle e\rangle}\,B_t^{\chi,\langle e\rangle}\right).$$

Hence, by Girsanov's theorem, if we define $B_t^{r,\langle e\rangle}$ such that:

$$\mathrm{d}\tilde{B}_t^{r,\langle e\rangle} = \mathrm{d}B_t^{r,\langle e\rangle} - \sigma^{\chi,\langle e\rangle}\mathrm{d}\langle B^{r,\langle e\rangle}, B^{\chi,\langle e\rangle}\rangle_t,$$

then $B_t^{r,\langle e\rangle}$ is a $\mathbb{Q}^{\langle 0\rangle}$ Brownian motion. In particular, assuming $\mathrm{d}\langle B^{r,\langle e\rangle}, B^{\chi,\langle e\rangle}\rangle_t = \rho^{\langle e\rangle}\mathrm{d}t$, we get the following $\mathbb{Q}^{\langle 0\rangle}$ dynamics for the short-rate of economy e:

$$\mathrm{d}r_t^{\langle e\rangle} = (a^{\langle e\rangle}\,(b^{\langle e\rangle} - r_t^{\langle e\rangle}) - \rho^{\langle e\rangle}\,\sigma^{\chi,\langle e\rangle})\mathrm{d}t + \sigma^{r,\langle e\rangle}\mathrm{d}B_t^{r,\langle e\rangle}.$$

For every counterparty c, the default time $\tau^{\langle c\rangle}$ can be modeled as a the stopping time $\inf\{t>0; \int_0^t \gamma_s^{\langle c\rangle}\mathrm{d}s \geq \epsilon^{\langle c\rangle}\}$, where $\epsilon^{\langle c\rangle}$ is a standard exponential. That is, for every $t\geq 0$,

$$\mathbb{1}_{\{\tau^{\langle c\rangle}\leq t\}} = 1 \Leftrightarrow \tau^{\langle c\rangle} \leq t \Leftrightarrow \int_0^t \gamma_s^{\langle c\rangle}\mathrm{d}s \geq \epsilon^{\langle c\rangle}. \tag{2.39}$$

For the instruments, we assume a book comprised of interest rate swaps at par at inception. For each swap, we denote the set of its reset dates by $\mathcal{R}$ and by t_- and t_+ the reset dates respectively immediately preceding and following t. We assume that successive reset dates are regularly spaced by δ, that the swap is spot starting, i.e. $0 \in \mathcal{R}$, and that the swap is paying fixed $\delta\, s$, where s is the swap rate, and receiving floating $\frac{1}{\mathrm{ZC}_{t_-}(t)} - 1$, where $\mathrm{ZC}_t(t')$ is the price of a zero-coupon bond[2.11] at time t with maturity t', at each reset date $t \in \mathcal{R} \setminus \{0\}$. Denoting by P_t^{sw} the price of the swap at time t in units of the underlying currency[2.12], we have for all $t \leq \bar{t} := \max \mathcal{R}$:

$$P_t^{sw} = \begin{cases} \dfrac{\mathrm{ZC}_t(t_+)}{\mathrm{ZC}_{t_-}(t_+)} - \mathrm{ZC}_t(\bar{t}) - \delta\, s \displaystyle\sum_{t' \in \mathcal{R}, t' > t} \mathrm{ZC}_t(t') & \text{if } t \notin \mathcal{R} \setminus \{0\} \\ \dfrac{1}{\mathrm{ZC}_{t_-}(t)} - \mathrm{ZC}_t(\bar{t}) - \delta\, s \,(1 + \displaystyle\sum_{t' \in \mathcal{R}, t' > t} \mathrm{ZC}_t(t')) & \text{if } t \in \mathcal{R} \setminus \{0\} \\ 1 - \mathrm{ZC}_0(\bar{t}) - \delta\, s \displaystyle\sum_{t' \in \mathcal{R} \setminus \{0\}} \mathrm{ZC}_0(t') & \text{if } t = 0 \end{cases}$$

Remark 2.13. The path-dependence induced by the previous reset date can be resorbed by including the short rates of that date among the risk factors.

2.11. Note that the price of a zero-coupon bond has a closed-form under our affine short-rate model.

2.12. The swap prices are then to be multiplied by the cross-currency exchange rate processes to have all prices in the same reference currency.

Chapter 3

Learning Value-at-Risk and Expected Shortfall

This chapter, also submitted as a paper, was co-authored with David Barrera, Stéphane Crépey, Emmanuel Gobet and Hoang-Dung Nguyen.

In a recent paper, Dimitriadis and Bayer devised a linear regression based method to jointly learn a conditional VaR (Value-at-Risk) and ES (Expected Shortfall) based on a joint elicitability representation developed by Fissler and Ziegel, and provided an asymptotic analysis of its convergence. We propose a non-asymptotic convergence analysis of an alternative two-steps approach to learn these functionals in a non-parametric setting using Rademacher and VC-based bounds. Our approach for the VaR is extended to the problem of learning multiple VaRs corresponding to multiple quantile levels. We provide efficient learning schemes based on neural network quantile and least-squares regressions and extensive numerical experiments in a Gaussian toy-model and a financial case-study where the objective is to learn a Dynamic Initial Margin (DIM).

Python notebooks reproducing the results of this paper are available at `https://github.com/BouazzaSE/Learning-VaR-and-ES`. HTML versions of the same notebooks are also available in order to view the experiments and the results on a browser without having to install Jupyter Notebook. Note that, due to GitHub size limitations, the HTML files must be downloaded locally (and then opened with a browser) to be displayed.

3.1 Introduction

Quantile regression is a classical statistical problem that has received attention since the 1750s. Recent developments are extensively surveyed in [Koenker, 2017], where it is noted that the least absolute criterion (or pinball loss function) for the median even preceded the least squares for the mean (introduced by Legendre in 1805).

Quantile regression is commonly done in the context of linear models, where the minimization problem can be cast as a linear program and subsequently solved using a simplex method. When several quantile levels are jointly considered, a flaw inherent to linear quantile regression is the problem of crossing quantile curves. Alternative approaches include nonlinear quantile regression based on interior point methods [Koenker and Park, 1996], or nonparametric quantile regression based on stochastic gradient descent methods [Rodrigues and Pereira, 2020].

Dimitriadis and Bayer, 2019 develop an asymptotic convergence analysis, establishing the consistency and asymptotic normality, under somewhat strong semiparametric assumptions and regularity conditions, of a joint linear regression estimator for the value-at-risk and expected shortfall based on their joint elicitability representation [Fissler and Ziegel, 2016; Fissler et al., 2016], which is implemented numerically using the Nelder-Mead optimization algorithm. Closer to our proposals, [Padilla et al., 2020] consider quantile regression with ReLU networks, including a discussion on minimax rates for quantile functions with Hölder-related regularity conditions, and providing qualitative non-asymptotic estimates for such networks, of which our corresponding results can be considered quantitative versions. [Shen et al., 2021] consider a different approach to the non-asymptotic analysis where they assume that the target conditional quantile function has a compositional structure using Hölder-continuous functions. The authors derive VC-based error bounds that depend on a combination of input dimension and the dimension of the composed functions, as opposed to only on the dimension of the inputs as usually found in the literature, and are therefore less impacted by the curse of dimensionality

In harmony with the numerous financial applications, we also refer to quantiles as value-at-risk (VaR). The contribution of the present paper is the non-asymptotic convergence analysis of a learning algorithm for the VaR and for the related expected shortfall (ES), i.e. the expected loss given the loss exceeds the VaR, possibly for multiple quantile levels at the same time, in a nonparametric setup. We also provide extensive discussions about practical implementations using feedforward neural networks, numerical experiments in a Gaussian toy model, and a financial case-study where the goal is to learn a dynamic initial margin in a multi-factor model.

The paper is outlined as follows: Section 3.2 presents our base learning algorithm. The convergence analysis of this scheme is performed in Section 3.3, relying on the general results of [Barrera, 2022]. We introduce multi-quantile extensions of the above scheme in Section 3.3.5. Appendix 3.A gathers classical elicitability results underlying different possible VaR and ES learning algorithms (including the one in Section 3.2, but also a joint representation à la [Fissler and Ziegel, 2016; Fissler et al., 2016], shown less efficient numerically in the paper's github).

We denote by $(\Omega, \mathcal{A}, \mathbb{P})$ a probability space, which admits all the random variables appearing below (the existence of $(\Omega, \mathcal{A}, \mathbb{P})$ can be verified a posteriori), with corresponding expectation operator denoted by $\mathbb{E}$, and we denote by $\mathcal{R}$ the Borel sigma algebra on $\mathbb{R}$.

3.2 A learning algorithm for VaR and ES

Let S be a Polish space with Borel sigma algebra $\mathcal{S}$. From now on

$$(X, Y) \colon \Omega \to S \times \mathbb{R}$$

is a fixed random vector in $S \times \mathbb{R}$[3.1], with $Y \in L^1_{\mathbb{P}}$. We will use the usual notation $\mathbb{P}_X$, $\mathbb{P}_{(X,Y)}$ for the laws of X and (X, Y): for every Borel sets $A \subset S$, $A' \subset S \times \mathbb{R}$

$$\mathbb{P}_X(A) = \mathbb{P}[X \in A], \qquad \mathbb{P}_{(X,Y)}(A') = \mathbb{P}[(X, Y) \in A']. \tag{3.1}$$

We fix a conditional distribution function $\mu \colon S \times \mathcal{R} \to [0, 1]$ of Y given X [Kallenberg, 2006], and we assume that the function $S \times \mathbb{R} \to \mathbb{R}$ defined by $(x, y) \mapsto \mu(x, (-\infty, y])$ is $(\mathcal{S} \otimes \mathcal{R})/\mathcal{R}$ (i.e. Borel)-measurable.[3.2] With these conventions, we will use implicitly the corresponding version

$$\mathbb{P}[Y \in \cdot | X] =: \mu(X, \cdot)$$

3.1. i.e. an $\mathcal{A}/(\mathcal{S} \otimes \mathcal{R})$ measurable function.

3.2. This the case if for instance $S = \mathbb{R}^d$ for some d and (X, Y) admits a density with respect to Lebesgue measure.

of the conditional probability of Y given X. In particular, we will use the conditional (cumulative) distribution (function) of Y given X,

$$F_{Y|X}(y) := \mathbb{P}[Y \le y \,|\, X] := \mu(X, (-\infty, y]).$$

We will finally assume, without loss of generality, that $F_{Y|X(\omega)}(\cdot)$ is integrable for every $\omega \in \Omega$.[3.3]

Definition 3.1. *The conditional value-at-risk (*VaR*) and expected shortfall (*ES*) of Y given X at the confidence level $\alpha \in (0,1)$ are (cf. (3.68))*

$$\begin{aligned} \mathrm{VaR}(Y|X) &= \mathrm{VaR}(F_{Y|X}) = \inf F^{-1}_{Y|X}([\alpha,1]) = \inf\{y \in \mathbb{R};\ F_{Y|X}(y) \ge \alpha\}, \\ \mathrm{ES}(Y|X) &= \frac{1}{1 - F_{Y|X}(\mathrm{VaR}(Y|X)-)} \int_{[\mathrm{VaR}(Y|X),\infty)} y\ F_{Y|X}(dy). \end{aligned} \tag{3.2}$$

Lemma 3.2. *The functions $\omega \mapsto \mathrm{VaR}(Y|X(\omega))$ and $\omega \mapsto \mathrm{ES}(Y|X(\omega))$ are $\sigma(X)$-measurable.*

Proof. Given $t \in \mathbb{R}$,

$$\{\mathrm{VaR}(Y|X) < t\} = \bigcup_{n \in \mathbb{N}^\star} \{F_{Y|X}(t - 1/n) \ge \alpha\},$$

which is a countable union of $\sigma(X)$-measurable sets ($F_{Y|X}(y)$ is $\sigma(X)$-measurable for every fixed y). This shows the claim for $\mathrm{VaR}(Y|X)$. As for the $\sigma(X)$-measurability of $\mathrm{ES}(Y|X)$, notice that the function $\mathrm{es}\colon S \times \mathbb{R} \to \mathbb{R}$ defined by

$$\mathrm{es}(v,x) = \frac{1}{1 - \mu(x, (-\infty, v))} \int y\, \mathbb{1}_{[v,\infty)}(y)\ \mu(x, dy)$$

is Borel-measurable (on the set where $\mu((-\infty, v), x) < 1$) and that

$$\mathrm{ES}(Y|X) = \mathrm{es}(X, \mathrm{VaR}(Y|X)).$$

□

From the Doob-Dynkin lemma, it follows that

Corollary 3.3. *There exist Borel measurable functions $q\colon S \to \mathbb{R}$ and $s\colon S \to \mathbb{R}$ such that*

$$q(X) = \mathrm{VaR}(Y|X), \qquad s(X) = \mathrm{ES}(Y|X), \qquad \mathbb{P}\text{-}a.\,s.$$

The goal of the article is to present and analyze algorithms for approximating ($\mathbb{P}_X$-versions of) the functions $q(\cdot)$ and/or $s(\cdot)$, efficient in high dimension d, based on i.i.d. samples of $(X, Y) \in \mathbb{R}^d \times \mathbb{R}$ and on suitable hypothesis spaces (including families of functions represented in terms of neural nets which are used in the experimental part of the paper)

$$\mathcal{F} = \{f\colon \mathbb{R}^d \to \mathbb{R}\}, \mathcal{G} = \{g\colon \mathbb{R}^d \to \mathbb{R}\}, \mathcal{H} = \{h = (f, g)\colon \mathbb{R}^d \to \mathbb{R}^2\}$$

for $q(\cdot)$, $s(\cdot)$, and $(q(\cdot), s(\cdot))$, respectively.

3.3. Since $Y \in L^1_{\mathbb{P}}$, we have that $\infty > \mathbb{E}[|Y|] = \mathbb{E}[\mathbb{E}[|Y|\,|X]] = \mathbb{E}[\int_{\mathbb{R}} |y|\, F_{Y|X}(dy)]$, thus $F_{Y|X(\omega)}$ is integrable for $\mathbb{P}$-a.e. ω: it suffices to change the version of X to guarantee integrability for every ω.

3.2.1 VaR and ES as optimization problems

To provide suitable representations of $\mathrm{VaR}(Y|X)$, $\mathrm{ES}(Y|X)$ in the context of convex optimisation, we will work under the following assumption.

Assumption 3.4. *$F_{Y|X}$ (defined by (3.2) for a given α) satisfies Assumption 3.35, $\mathbb{P}$-a.s., and if Λ_f, φ_g, $\Lambda_{f,g}$, $q(\cdot)$ and $s(\cdot)$ are respectively as in (3.73), (3.74), (3.75), (3.3), then $\Lambda_f(Y, q(X))$, $\varphi_g(Y, q(X), s(X) - q(X))$ and $\Lambda_{f,g}(Y, q(X), s(X))$ are $\mathbb{P}$-integrable.*

Our methods rely on the following functional representations of $\mathrm{VaR}(Y|X)$ and $\mathrm{ES}(Y|X)$. We use implicitly in the statement the convention $\mathbb{E}[h(X,Y)] = \infty$ whenever $h(X,Y)$ is not $\mathbb{P}$-integrable. We also use the notation $\mathcal{L}(S)$ [resp. $\mathcal{L}_+(S)$] for the space of Borel measurable functions $S \to \mathbb{R}$ [resp. $S \to \mathbb{R}_+$].

Theorem 3.5. *Under Assumption 3.4:*

$$q(\cdot) \in \operatorname*{argmin}_{f \in \mathcal{L}(S)} \mathbb{E}[\Lambda_f(Y, f(X))], \tag{3.3}$$

$$s(\cdot) - q(\cdot) \in \operatorname*{argmin}_{g \in \mathcal{L}_+(S)} \mathbb{E}[\varphi_g(Y, q(X), g(X))], \tag{3.4}$$

$$(q(\cdot), s(.)) \in \operatorname*{argmin}_{(f,g) \in \mathcal{L}(S) \times \mathcal{L}(S)} \mathbb{E}[\Lambda_{f,g}(Y, f(X), g(X))]. \tag{3.5}$$

If $(Y - q(X))^+$ is $\mathbb{P}$-integrable, then

$$s(X) = q(X) + (1-\alpha)^{-1}\, \mathbb{E}[(Y - q(X))^+ | X], \qquad \mathbb{P}\text{-}a.\,s. \tag{3.6}$$

(this does not depend on the assumptions on Λ_f, φ_g, $\Lambda_{f,g}$).

Proof.

All these statements are a straightforward consequence of the fact that, if $h(X,Y)$ is $\mathbb{P}$-integrable, then

$$\mathbb{E}[h(X,Y)|X] = \int_{\mathbb{R}} h(X, y)\, F_{Y|X}(dy), \qquad \mathbb{P}\text{-}a.\,s.,$$

together with the characterizations of VaR and ES in Lemmas 3.37, 3.38 and 3.39.

To illustrate for $q(X)(\cdot)$: using Lemma 3.37 and the above identity, we obtain that

$$\mathbb{E}[\Lambda_f(Y, q(X))|X] \le \mathbb{E}[\Lambda_f(Y, f(X))|X], \qquad \mathbb{P}\text{-}a.\,s., \tag{3.7}$$

for every $f \in \mathcal{L}(S)$. This implies (3.3) by integrating with respect to $\mathbb{P}$. The other statements can be proved in a similar fashion. □

Remark 3.6. If $(Y - q(X))^+ \in L^2_{\mathbb{P}}$, then the representation (3.6) is also a consequence of the characterization (3.4). To see this notice that, by the Pythagorean theorem and the nonnegativity of $(Y - q(X))^+$, any $r \in \mathcal{L}_+(S)$ satisfying

$$r(\cdot) \in \operatorname*{argmin}_{g \in \mathcal{L}_+(S)} \mathbb{E}[((1-\alpha)^{-1}(Y - q(X))^+ - g(X))^2] \tag{3.8}$$

has the property that

$$r(X) = \mathbb{E}[(1-\alpha)^{-1}(Y - q(X))^+ | X], \qquad \mathbb{P}\text{-}a.\,s. \tag{3.9}$$

Notice next that, with the choice $g(z) = z^2$, φ_g in (3.74) is given by

$$\varphi_g(y, v, z) = z^2 - 2(1-\alpha)^{-1}(y - v)^+ z \tag{3.10}$$

$$= (z - (1-\alpha)^{-1}(y - v)^+)^2 - ((1-\alpha)^{-1}(y - v)^+)^2. \tag{3.11}$$

from where it follows that the minimization criteria (3.4) and (3.8) are then exactly the same, leading in particular to

$$s(X) - q(X) = r(X) = (1-\alpha)^{-1}\mathbb{E}[(Y - q(X)(X))^{+}|X], \qquad \mathbb{P}\text{-}a.\,s., \tag{3.12}$$

as claimed by (3.6).

Remark 3.7. The minimizers in (3.3)-(3.5) do not need to be unique: notice for instance that the proof of (3.3) (illustrated above) shows that any function $q(X)_1 : S \to \mathbb{R}$ satisfying $F_{Y|X}(q(X)_1(X)) = \alpha$ is a minimizer of $f \mapsto \mathbb{E}[\Lambda_f(Y, f(X))]$, and that there are infinitely many such functions if $F^{-1}_{Y|X}(\alpha)$ is an interval with positive length, $\mathbb{P}$-a.s.

3.2.2 The algorithm

The functional representations in (3.3)-(3.8) give immediately rise to equally many approximation algorithms for conditional VaR and/or ES. In all cases, the numerical recipe is simply that of replacing the minimization problems in (3.3)-(3.8) by empirical versions. Instead of $\mathcal{L}(S), \mathcal{L}_+(S)$ and $\mathcal{L}(S) \times \mathcal{L}(S)$ we use convenient hypotheses spaces $\mathcal{F} \subset \mathcal{L}(S)$, $\mathcal{G} \subset \mathcal{L}_+(S)$, and $\mathcal{H} \subset \mathcal{L}(S) \times \mathcal{L}(S)$. Instead of integration with respect to $\mathbb{P}$ we use a Monte Carlo approximation based on (properly truncated) i.i.d. samples of (X, Y).

After some preliminary empirical investigations reported in the paper's GitHub, the best turned out to be the simplest, i.e. the two-steps algorithm that first uses (3.3) to obtain an approximation $\hat{q}(\cdot)$ of the (conditional) VaR, and then uses (3.6) together with the interpretation of the conditional expectation as a least-squares minimization problem, i.e. (3.8), to learn ES, using the approximation $\hat{q}(\cdot)$ obtained before. This two-steps algorithm will be our main focus in what follows. Its pseudo-code is provided as Algorithm 3.1. The restrictions on $\mathcal{F}$ and f, the transformation h_1, h_2 and the truncations T_B defined by

$$T_B\, y = \max\{\min\{y, B\}, -B\}, \qquad (y, B) \in \mathbb{R} \times [0, \infty) \tag{3.13}$$

permit a fitting of the algorithm within the framework of the bounds developed in [Barrera, 2022]. They may also have practical advantages, as discussed in Appendix 3.B.

Algorithm 3.1
Estimating the conditional VaR and ES by regression in two steps

Parameters:

- The loss Λ given by (3.73) with $\iota(z) = z$.
- Constants $(B_1, B_2, B_3) \in (0, \infty)^3$ with $B_1 \leq B_2$.
- A function $h_1 : S \times \mathbb{R} \to [-B_2, B_2]$ such that, for $\mathbb{P}_X$-a.e. $x \in S$, $h_{1,x}(\cdot) := h_1(x, \cdot)$ is increasing in a set I_x with $\mathbb{P}(Y \in I_x | X = x) = 1$.
- A conditionally affine function $h_2(x, y) = \tau(x)\, y + \nu(x)$ with $\tau(x) > 0$ for $\mathbb{P}_X$ a.e. $x \in S$.
- A set $\mathcal{F}$ of Borel measurable functions $S \to [-B_1, B_1]$.
- A set $\mathcal{G}$ of Borel measurable functions $S \to [0, B_3]$.

Input: An i.i.d sample $D = \{(X_k, Y_k)\}_{k=1}^n$ of (X, Y).

Compute $\hat{f} \in \operatorname{argmin}_{f \in \mathcal{F}} \sum_{k=1}^n \ell(h_1(X_k, Y_k), f(X_k))$;

Set $\hat{q}(x) = h^{-1}_{1,x} \circ \hat{f}(x)$;

Compute $\hat{g} \in \operatorname{argmin}_{g \in \mathcal{G}} \sum_{k=1}^n (g(X_k) - T_{B_3}((1-\alpha)^{-1}(h_2(X_k, Y_k) - h_2(X_k, \hat{q}(X_k))^{+}))^2$;

Set $\hat{r}(x) = (\hat{g}(x) - \nu(x))\,/\,\tau(x)$;

Return $(\widehat{\mathrm{VaR}}(Y|.), \widehat{\mathrm{ES}}(Y|.)) = (\hat{q}(.), \hat{q}(.) + \hat{r}\,(.))$

3.3 Convergence Analysis of the Learning Algorithm

In what follows, we will be using the assumption $h_k(x, y) = y$ $(k = 1, 2)$ for the data transformations in Algorithm 3.1. Our results, therefore, leave open the error induced by the operations $(h_k(X, \cdot))^{-1}$ used for the final estimates.

We will use the notation

$$D = \{(X_j, Y_j)\}_{j=1}^{n} \tag{3.14}$$

for an i.i.d. sample of (X, Y) (with n given). If there are several (say l) of these samples, we will denote them by

$$D_k = \{(X_{k,j}, Y_{k,j})\}_{j=1}^{n_k}, \qquad k = 1, \ldots, l. \tag{3.15}$$

Using also the notation (3.73), (3.74), (3.75), we will denote, for $(f, g) \in \mathcal{L}(S) \times \mathcal{L}_+(S)$

$$\begin{aligned}
\tilde{\Lambda}_\iota(f) &= \mathbb{E}[\Lambda_\iota(Y, f(X))], \\
\hat{\Lambda}_\iota(f) &= \frac{1}{n}\sum_{k=1}^{n} \Lambda_\iota(Y_k, f(X_k)), \\
\tilde{\varphi}_\varsigma(f, g) &= \mathbb{E}[\varphi_\varsigma(Y, f(X), g(X))], \\
\hat{\varphi}_\varsigma(f, g) &= \frac{1}{n}\sum_{k=1}^{n} \varphi_\varsigma(Y_k, f(X_k), g(X_k)), \\
\tilde{\Lambda}_{\iota,\varsigma}(f, g) &= \mathbb{E}[\Lambda_{\iota,\varsigma}(Y, f(X), f(X) + g(X))], \\
\hat{\Lambda}_{\iota,\varsigma}(f, g) &= \frac{1}{n}\sum_{k=1}^{n} \Lambda_{\iota,\varsigma}(Y_k, f(X_k), f(X_k) + g(X_k)).
\end{aligned} \tag{3.16}$$

Throughout this section,

$$\mathcal{F} \subset \mathcal{L}(S), \qquad \mathcal{G} \subset \mathcal{L}_+(S), \qquad \mathcal{H} \subset \mathcal{L}(S) \times \mathcal{L}_+(S) \tag{3.17}$$

will be fixed hypothesis spaces. Associated to these and to the loss functions in (3.15) there are the following quantities of interest,

$$\tilde{q} \in \operatorname*{argmin}_{f \in \mathcal{F}} \tilde{\Lambda}_\iota(f), \qquad \hat{q} \in \operatorname*{argmin}_{f \in \mathcal{F}} \hat{\Lambda}_\iota(f) \tag{3.18}$$

and given $f \in \mathcal{L}(S)$,

$$\tilde{r}_f \in \operatorname*{argmin}_{g \in \mathcal{G}} \tilde{\varphi}_\varsigma(f, g), \qquad \hat{r}_f \in \operatorname*{argmin}_{g \in \mathcal{G}} \hat{\varphi}_\varsigma(f, g). \tag{3.19}$$

Thus (3.18) defines respectively *the best mean and empirical hypothesis for* VaR *within* $\mathcal{F}$, and (3.19) defines *the best mean and empirical hypotheses for* $\mathrm{ES} - \mathrm{VaR}$ *within* $\mathcal{G}$ *conditioned to the hypothesis* f *for* VaR (f may not belong to $\mathcal{F}$). Similarly, we define the *best mean and empirical joint hypotheses for* $(\mathrm{VaR}, \mathrm{ES} - \mathrm{VaR})$ respectively by

$$(\tilde{q}, \tilde{r}) \in \operatorname*{argmin}_{h=(f,g) \in \mathcal{H}} \tilde{\Lambda}_{\iota,\varsigma}(f, g), \qquad (\hat{q}, \hat{r}) \in \operatorname*{argmin}_{h=(f,g) \in \mathcal{H}} \hat{\Lambda}_{\iota,\varsigma}(f, g). \tag{3.20}$$

3.3.1 The approximation error of the estimator of VaR

Algorithm 3.1 is based on the following assumption:

Assumption 3.8. *The function* $f: \mathbb{R} \to \mathbb{R}$ *in (3.73) is the identity function. We therefore omit* f *and write*

$$\Lambda(y, v) = (1 - \alpha)^{-1} (y - v)^+ + v,$$

as well as $\tilde{\Lambda}(\cdot)$ *and* $\hat{\Lambda}(\cdot)$ *instead of* $\Lambda_\iota(\cdot,\cdot), \tilde{\Lambda}_\iota(\cdot)$ *and* $\hat{\Lambda}_\iota(\cdot)$.

Assumption 3.8 implies the convexity of $\Lambda(y,\cdot)$ (for all y), which we exploit in several manners. In a sense, Assumption 3.8 is only an apparent restriction: notice that for any $(y,v)\in\mathbb{R}^2$

$$\Lambda_\iota(y,v)=\Lambda(\iota(y),\iota(v)),$$

which allows us to transport any conclusion under Assumption 3.8 to the respective conclusion for generic ι, by "transferring" the hypotheses related to (y,v) to hypotheses related to $(\iota(y),\iota(v))$.

The following assumption is a conditional version of Assumption 3.35:

Assumption 3.9. *There exist functions* $a,b\colon S\to\mathbb{R}$ *such that*

$$F_{Y|X}(a(X))<\alpha\le F_{Y|X}(b(X)), \tag{3.21}$$

on a set Ω_0 *of* $\mathbb{P}$*-measure one and such that* $F_{Y|X(\omega)}(\cdot)$ *is absolutely continuous in* $[a(X(\omega)),b(X(\omega))]$ *for every* $\omega\in\Omega_0$.

Notice that, under this assumption, $a(X)\le q(X)\le b(X)$ except on a set of measure zero.

Assumption 3.10. *(for a generic family* $\mathcal{F}_1\subset\mathcal{L}(S)$*) Assumption 3.9 holds, and* $\mathbb{F}_1\subset\mathcal{L}(S)$ *is such that*

1. *For every* $f\in\mathcal{F}_1$, $a(X)\le f(X)\le b(X)$, *except on a set* Ω_0 *of* $\mathbb{P}$*-measure zero.*
2. *There exists* $c_{\mathcal{F}_1}>0$ *such that, for every* $f\in\mathcal{F}_1$,

$$F'_{Y|X}(f(X))\ge c_{\mathcal{F}_1}\quad,\qquad \mathbb{P}\text{-}a.\,s.$$

Assumption 3.10 is needed to succeed in applying Taylor expansions towards the estimation of errors in our analysis.

Lemma 3.11. *Given* $\mathcal{F}\subset\mathcal{L}(S)$, *and under Assumption 3.8, define* $\tilde{q}$ *by (3.18), let* $\mathcal{F}_0\subset\mathcal{F}$, *and consider*

$$\mathcal{F}_0^*=\{t\,f+(1-t)\,q\colon (t,f)\in[0,1]\times\mathcal{F}_0\},\qquad \mathcal{F}^*=\{t\,f+(1-t)\,q\colon (t,f)\in[0,1]\times\mathcal{F}\},$$

If $\mathcal{F}_1\equiv\mathcal{F}^*$ *satisfies Assumption 3.10 and if*

$$C_{\mathcal{F}_0^*}=\sup_{f\in\mathcal{F}_0^*}\{||\,F'_{Y|X}(f(X))||_{\mathbb{P},\infty}\},$$

then the inequalities

$$\begin{aligned} c_{\mathcal{F}^*}||\tilde{q}-q||^2_{\mathbb{P}_X,2} &\le 2\,(1-\alpha)\,(\tilde{\Lambda}(\tilde{q})-\tilde{\Lambda}(q))\\ &\le (2\,(2-\alpha)\inf_{f\in\mathcal{F}}||\,f-q||_{\mathbb{P}_X,1})\wedge(C_{\mathcal{F}_0^*}\inf_{f\in\mathcal{F}_0}||\,f-q||^2_{\mathbb{P}_X,2}) \end{aligned} \tag{3.22}$$

hold.

Proof. For any $f\in\mathcal{F}$, consider the function $[0,1]\to\mathbb{R}$ defined by

$$t\mapsto V_f(t)=\tilde{\Lambda}(q+t\,(f-q)),$$

which has a minimum at $t=0$.

We use the definition of $F_{Y|X}(\cdot)$ and differentiation under the integral sign to obtain, for every $t \in [0,1]$

$$\begin{aligned} V_f''(t) &= \frac{\partial^2}{\partial t^2}\mathbb{E}\left[\int_{\mathbb{R}} \Lambda(y, q(X) + t\,(f(X) - q(X)))\, F_{Y|X}\,(d\,y)\right] \\ &= \frac{\partial}{\partial t}\mathbb{E}[(f(X) - q(X))\,((1-\alpha)^{-1}\,(F_{Y|X}\,(q(X) + t\,(f(X) - q(X))) - 1) + 1)] \\ &= \mathbb{E}[(f(X) - q(X))^2\, F'_{Y|X}\,(q(X) + t\,(f(X) - q(X)))\,/\,(1-\alpha)] \\ &\geq \frac{c_{\mathcal{F}^*}}{1-\alpha}\mathbb{E}[(f(X) - q(X))^2]. \end{aligned} \tag{3.23}$$

This shows in particular that V_f is twice continuously differentiable (from the right at $t=0$) and convex. Applying Taylor's theorem and the fact that $V_f'(0) = 0$ we arrive at

$$\frac{c_{\mathcal{F}^*}}{2\,(1-\alpha)}||\,f - q\,||^2_{\mathbb{P}_X,2} \leq \tilde{\Lambda}(f) - \tilde{\Lambda}(q). \tag{3.24}$$

Since this is valid for any $f \in \mathcal{F}$, it is valid for $f = \tilde{q}$. This gives

$$\frac{c_{\mathcal{F}^*}}{2\,(1-\alpha)}||\,\tilde{q} - q\,||^2_{\mathbb{P}_X,2} \leq \tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q). \tag{3.25}$$

The upper bound

$$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \frac{C_{\mathcal{F}_0^*}}{2\,(1-\alpha)} \inf_{f \in \mathcal{F}_0} ||\,f - q\,||^2_{\mathbb{P}_X,2} \tag{3.26}$$

follows from the inequality $\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \tilde{\Lambda}(f) - \tilde{\Lambda}(q)$ (valid for any $f \in \mathcal{F}_0$) and an obvious modification of the previous argument starting from the last inequality in (3.22).

Finally, the upper bound

$$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \left(\frac{2-\alpha}{1-\alpha}\right) \inf_{f \in \mathbb{F}} ||\,f - q\,||_{\mathbb{P}_X,1} \tag{3.27}$$

follows via an elementary estimation using

$$|a^+ - b^+| \leq |a - b| \tag{3.28}$$

and the triangle inequality, together (again) with the inequality $\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \tilde{\Lambda}(f) - \tilde{\Lambda}(q)$, valid for every $f \in \mathcal{F}$. The conclusion follows from (3.25), (3.26) and (3.27). □

Remark 3.12. Notice that as $\mathcal{F}_0$ gets larger, $C_{\mathcal{F}_0^*}$ in (3.11) increases and $\inf_{f \in \mathbb{F}_0} ||\,f - q\,||_{\mathbb{P}_X,2}$ decreases: by making the bound (3.11) depend of $\mathcal{F}_0 \subset \mathcal{F}$ we leave open the room for a trade-off between these quantities.

Remark 3.13. If we strengthen Assumption 3.10 by requiring that for some $(c, C) \in (0,\infty) \times (0,\infty)$, and except on a set of $\mathbb{P}$-measure zero

$$c \leq F'_{Y|X}(y) \leq C \quad , \quad \text{for every } y \in [a(X), b(X)], \tag{3.29}$$

then the conclusion of Lemma 3.11 holds with $(c_{\mathcal{F}^*}, C_{\mathcal{F}_0^*})$ replaced by (c, C) under the sole assumption that, for every $f \in \mathcal{F}$,[3.4]

$$[f(X), q(X)] \cup [q(X), f(X)] \subset [a(X), b(X)], \qquad \text{except on a set of } \mathbb{P}\text{-measure zero.} \tag{3.30}$$

3.4. $[u,v] \cup [v,u]$ is just the closed segment of the real line determined by $(u,v) \in \mathbb{R}^2$. Notice that (3.30) is exactly the same as 1. in Assumption 3.10 for $\mathcal{F}_1 = \mathcal{F}^*$.

As will be illustrated in Examples 3.14 and 3.15, these observations allow weakening the dependence on $\mathcal{F}$ in the estimate (3.11).

Example 3.14. Assume (3.29) and, given $\delta > 0$, assume that $\mathcal{F}$ is such that (3.30) holds and

$$\inf_{f \in \mathcal{F}} || f - q||_{\mathbb{P}_X,2} < \delta.$$

Denoting by $\tilde{q}_\delta$ the solution to the left-hand side of (3.18), an application of Remark 3.13 gives that

$$c||\tilde{q}_\delta - q||^2_{\mathbb{P}_X,2} \leq \delta\,(2\,(2-\alpha) \wedge C\,\delta) \leq C\,\delta^2,$$

leading to the estimate

$$||\tilde{q}_\delta - q||_{\mathbb{P}_X,2} \leq \left(\frac{C}{c}\right)^{1/2} \delta.$$

Example 3.15. To give a concrete instance of the previous example, assume that, for some $(A, B) \in \mathbb{R}^2$,

$$q(X) \in [A, B], \mathbb{P}_X\text{-}a.\,s.$$

(see also Remark 3.34), assume that (3.21) and (3.29) hold with $a(X) \equiv A$ and $b(X) \equiv B$, and assume that there exists a finite or countable partition $\{S_j\}_j \subset \mathcal{S}$ of S such that, for all j,

$$||q||_{TV_{S_j}} = \sup_{(x,x') \in S_j \times S_j} |q(x) - q(x')| < \delta$$

(for instance if q is continous, as S is a Polish space). Then (3.14) holds with

$$\mathcal{F} = \{x \mapsto \sum_j a_j\, \mathbb{1}_{S_j}(x) \colon a_j \in [A, B], \forall j\}.$$

Partitions $\{S_j\}_j$ as above can be available with only partial information on q on cases of interests: consider for example the case in which S is compact and q is uniformly Lipschitz with a known Lipschitz constant.

3.3.2 A confidence interval for the estimator of VaR

Let us now give an upper bound for the error in probability associated to the empirical estimator $\hat{q}$ of $\tilde{q}$. For this, we need to introduce the following measures of complexity applicable to the families of hypotheses used along our schemes:

Definition 3.16. *If S is a Polish space, $\mathcal{H} \subset \mathcal{L}(S)$, and $X_{1:n}$ is a random sequence in S, the empirical Rademacher complexity $\mathcal{R}_{\text{emp}}(\mathcal{H}, X_{1:n})$ and the Rademacher complexity $\mathcal{R}_{\text{ave}}(\mathcal{H}, X_{1:n})$ of $\mathcal{H}$ at $X_{1:n}$ are defined as*

$$\mathcal{R}_{\text{emp}}(\mathcal{H}, X_{1:n}) = \mathbb{E}\left[\sup_{h \in \mathcal{H}} \sum_{k=1}^{n} U_k\, h(X_k) \middle| X_{1:n}\right], \qquad \mathcal{R}_{\text{ave}}(\mathcal{H}, X_{1:n}) = \mathbb{E}[\mathcal{R}_{\text{emp}}(\mathcal{H}, X_{1:n})]$$

where $U_{1:n}$ is an i.i.d. Rademacher sequence $\mathbb{P}[U_k = 1] = \mathbb{P}[U_k = -1] = 1/2$ independent of $X_{1:n}$.

The Rademacher complexities have the following property, which we will use later and whose proof is an easy exercise: if

$$\text{co}(\mathcal{H}) = \bigcup_m \left\{\sum_{k=1}^{m} t_k\, h_k \colon h_{1:m} \in \mathcal{H}^m, t_{1:m} \in [0,1]^m, \sum_k t_k = 1\right\} \tag{3.31}$$

is the convex hull of $\mathcal{H}$, and if

$$\mathrm{cobal}(\mathcal{H}) = \mathrm{co}(\mathcal{H} \cup -\mathcal{H})$$

is the balanced convex hull of $\mathcal{H}$, then

$$\mathcal{R}_{\mathrm{emp}}(co(\mathcal{H}), X_{1:n}) = \mathcal{R}_{\mathrm{emp}}(\mathcal{H}, X_{1:n}), \qquad \mathcal{R}_{\mathrm{emp}}(cobal(\mathcal{H}), X_{1:n}) \leq 2\mathcal{R}_{\mathrm{emp}}(\mathcal{H}, X_{1:n}).$$

Definition 3.17. *If $S, \mathcal{H}$ and $X_{1:n}$ are as in Definition 3.16, and if $r \geq 0$, the covering number of $\mathcal{H}$ with respect to the empirical L^1-norm at $X_{1:n}$, $N_1(\mathcal{H}, X_{1:n}, r)$, is defined as*

$$N_1(\mathcal{H}, X_{1:n}, r) := \inf\left\{ m \in \mathbb{N} \colon \exists\, g_{1:m} \in \mathcal{L}^m(S) \colon \sup_{h \in \mathcal{H}} \min_{l} \sum_{k=1}^{n} |h(X_k) - g_l(X_k)| < n\, r \right\}; \tag{3.32}$$

with the convention $\inf \emptyset = \infty$. A sequence $g_{1:m}$ satisfying the condition in (3.32) is called an r-covering of $\mathcal{H}$ with respect to the empirical $L^1 -$ norm at $X_{1:n}$.

In what follows, $(X, Y)_{1:n}$ is the sample (3.14) used to compute $\hat{q}$ and

$$\Lambda(\mathcal{F}) = \{(x, y) \mapsto \Lambda(y, f(x)) \colon f \in \mathcal{F}\},$$

is the family of instantaneous losses associated to $\mathcal{F}$.

Lemma 3.18. *Under the hypotheses of Lemma 3.11, and given $\delta \in (0, 1)$, the bound*

$$\begin{aligned} c_{\mathcal{F}^*} || \hat{q} - q ||^2_{\mathbb{P}_X, 2} \;\leq\; & \left(2\,(2-\alpha) \inf_{f \in \mathcal{F}} || f - q ||_{\mathbb{P}_X, 1} \right) \wedge \left(C_{\mathbb{F}_0^*} \inf_{f \in \mathcal{F}_0} || f - q ||^2_{\mathbb{P}_X, 2} \right) \\ & + (1-\alpha) \left(\frac{2^5}{n} \right)^{1/2} \left(\sup_{f \in \mathcal{F}} ||\Lambda(Y, f(X))||_{\mathbb{P}, \infty} \left(\log\left(\frac{2}{\delta} \right) \right)^{1/2} \right) \\ & + \frac{8\,(1-\alpha)}{n} \mathcal{R}_{\mathrm{ave}}(\Lambda(\mathcal{F}), (X, Y)_{1:n}) \end{aligned} \tag{3.33}$$

holds with probability at least $1 - \delta$. The right-hand side of (3.18) can be further upper bounded via the inequalities, valid for every $r > 0$

$$\begin{aligned} \mathcal{R}_{\mathrm{ave}}(\Lambda(\mathcal{F}), D) \;&\leq\; ((2-\alpha)/(1-\alpha)) \mathcal{R}_{\mathrm{ave}}(\mathcal{F}, X_{1:n}) \\ &\leq\; ((2-\alpha)/(1-\alpha)) (r + \sqrt{n} \sup_{f \in \mathcal{F}} || f(X) ||_{\mathbb{P}, \infty} \mathbb{E}\big[\sqrt{2 \log(N_1(\mathcal{F}, X_{1:n}, r/n))} \big]). \end{aligned} \tag{3.34}$$

Remark 3.19. If $\max\{|| Y ||_{\mathbb{P},\infty}, \sup_{f \in \mathcal{F}} || f(X) ||_{\mathbb{P},\infty}\} \leq B$ then, clearly,

$$\sup_{f \in \mathcal{F}} ||\Lambda(Y, f(X))||_{\mathbb{P},\infty} \leq \left(\frac{2-\alpha}{1-\alpha} \right) B.$$

Proof. (of Lemma 3.18) According to (3.24), for every $f \in \mathcal{F}$

$$c_{\mathcal{F}^*} || f - q ||^2_{\mathbb{P}_X, 2} \leq 2\,(1-\alpha)\,(\tilde{\Lambda}(f) - \tilde{\Lambda}(q)),$$

implying in particular that

$$c_{\mathcal{F}^*} || \hat{q} - q ||^2_{\mathbb{P}_X, 2} \leq 2\,(1-\alpha)\,((\tilde{\Lambda}(\hat{q}) - \tilde{\Lambda}(\tilde{q})) + (\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q)))$$

The term $2(1-\alpha)(\tilde{\Lambda}(\tilde{q})-\tilde{\Lambda}(q))$ is upper bounded in (3.11). To upper bound $\tilde{\Lambda}(\hat{q})-\tilde{\Lambda}(\tilde{q})$ in probability we apply the Rademacher bound [Barrera, 2022] taking $Z_k=(X_k,Y_k)\sim(X_1, Y_1)$ i.i.d. and the diagonal family

$$\Lambda(\mathcal{F})_{1:n}^{(n)}=\{((x_k,y_k))_{k\in 1:n}\mapsto(\Lambda(f)(x_k,y_k)/n)_{k\in 1:n}\colon f\in\mathcal{F}\}$$

to obtain the inequality (see also [Barrera, 2022](eqns. (2.25), (2.26))

$$\begin{aligned}\tilde{\Lambda}(\hat{q})-\tilde{\Lambda}(\tilde{q}) &\leq 2\left((1/\sqrt{n})\sup_{f\in\mathcal{F}}||\Lambda(Y,f(X))||_{\mathbb{P},\infty}\sqrt{2\log(2/\delta)}+(2/n)\mathcal{R}_{\text{ave}}(\Lambda(\mathcal{F}),D)\right)\\ &= (2^3/n)^{1/2}\left(\sup_{f\in\mathcal{F}}||\Lambda(Y,f(X))||_{\mathbb{P},\infty}(\log(2/\delta))^{1/2}+(2/n)^{1/2}\mathcal{R}_{\text{ave}}(\Lambda(\mathcal{F}),D)\right)\end{aligned} \tag{3.35}$$

with probability at least $1-\delta$. We deduce (3.18) combining (3.19) with the above.To prove the first inequality in (3.33), note that by Talagrand contraction lemma [Mohri et al., 2018](Lemma 4.2 p.78), since $u\mapsto(1-\alpha)^{-1}u^+$ is $(1-\alpha)^{-1}$-Lipschitz, then for any $(x,y)_{1:n}\subset(S\times\mathbb{R})^n$

$$\begin{aligned}\mathcal{R}_{\text{emp}}(\Lambda(\mathcal{F}),(x,y)_{1:n}) &\leq \mathcal{R}_{\text{emp}}(\{(x,y)\mapsto(1-\alpha)^{-1}(y-f)^+\colon f\in\mathcal{F}\},(x,y)_{1:n})+\mathcal{R}_{\text{emp}}(\mathcal{F},x_{1:n})\\ &\leq (1-\alpha)^{-1}\mathcal{R}_{\text{emp}}(\{(x,y)\mapsto y-f\colon f\in\mathcal{F}\},(x,y)_{1:n})+\mathcal{R}_{\text{emp}}(\mathcal{F},x_{1:n})\\ &\leq (1-\alpha)^{-1}\underbrace{\mathcal{R}_{\text{emp}}(\{\text{Id}_{\mathbb{R}}\},y_{1:n})}_{=0}+\frac{2-\alpha}{1-\alpha}\mathcal{R}_{\text{emp}}(\mathcal{F},x_{1:n})\end{aligned}$$

which implies the first inequality in (3.33) by integration with respect to the law of D.

The second and third inequalities in (3.33) are a direct consequence of [Barrera, 2022](eqn. (3.47)) and the argument in [Barrera, 2022](eqn. (3.53))[Barrera, 2022]. The fourth follows easily from the fact that if $\mathcal{F}'\subset\mathcal{L}(S)$ is a $(1-\alpha)r/(2-\alpha)$ covering of $\mathcal{F}$ with respect to the empirical L^1-norm at $x_{1:n}$, then $\{(x,y)\mapsto y-f(x)|f\in\mathcal{F}'\}$ is an r-covering of $\Lambda(\mathcal{F})$ with respect to the empirical L^1-norm at $(x,y)_{1:n}$ (this can be proved using (3.28)). □

Let us now introduce the following hypothesis, which covers the estimation error of $\hat{f}$ in Algorithm 3.1.

Assumption 3.20. *For given $0<B_1\leq B_2$,*

$$||Y||_{\mathbb{P},\infty}\leq B_2.$$

In addition, $\text{VaR}(Y|X)$ *takes values in* $(-B_1,B_1]$ *and* $y\mapsto F_{Y|X(\omega)}[(-\infty,y]]$ *is* $\mathbb{P}$*-a.e. differentiable, with derivative uniformly bounded away from* 0 *and* ∞ *in* $[-B_1,B_1]$*. That is,*

$$F_{Y|X}(-B_1)<\alpha\leq F_{Y|X}(B_1),\qquad \mathbb{P}\text{-}a.\,s.,$$

and there exist $0<c_{B_1}\leq C_{B_1}<\infty$ *such that*

$$c_{B_1}\leq F'_{Y|X}(y)\leq C_{B_1},\qquad \mathbb{P}\text{-}a.\,s.,$$

for every $y\in[-B_1,B_1]$.

Using Assumption 3.20, the following result follows easily from Lemma 3.18:

Theorem 3.21. *Under Assumption 3.20, let*

$$\mathcal{F}'\subset\mathcal{F}\subset\text{co}(\mathcal{F}')\subset\mathcal{L}(S) \tag{3.36}$$

where $\mathcal{F}$ is a family of functions uniformly bounded by B_1 (see also (3.31)). Then the inequality

$$\begin{aligned} c_{B_1}||\hat{q}-q||^2_{\mathbb{P}_X,2} \;\le\; & \left(2\,(2-\alpha)\,\inf_{f\in\mathcal{F}}||\,f-q||_{\mathbb{P}_X,1}\right)\wedge\left(C_{B_1}\,\inf_{f\in\mathcal{F}}||\,f-q||^2_{\mathbb{P}_X,2}\right)\\ & +\frac{4\,(2-\alpha)\,B_2}{\sqrt{n}}\sqrt{2\log\left(\frac{2}{\delta}\right)}\\ & +\frac{8\,(2-\alpha)\,B_2}{\sqrt{n}}\left(1+\mathbb{E}\left[\sqrt{2\log\left(N_1\left(\mathcal{F}',X_{1:n},\frac{B_1}{\sqrt{n}}\right)\right)}\right]\right) \end{aligned} \tag{3.37}$$

holds or every $\delta\in(0,1)$ with probability at least $1-\delta$.

Proof. As discussed in Remark 3.13, it is easy to see that the hypotheses of Lemma 3.11 hold for $c_{\mathcal{F}^*}=c_{B_1}$ and $C_{\mathcal{F}^*}=C_{B_1}$ in this case.

The second inequalities in (3.33) and [Barrera, 2022](Remark 3.4) for $(\mathcal{H}'_{1:n},\mathcal{H}_{1:n})=(\mathrm{diag}(\mathcal{F}')_{1:n},\mathrm{diag}(\mathcal{F})_{1:n})$ (see [Barrera, 2022](eqn (2.3))) give that for every $\delta>0$

$$\mathcal{R}_{\mathrm{ave}}(\Lambda(\mathcal{F}),(X,Y)_{1:n})\le\frac{2-\alpha}{1-\alpha}(\delta+\sqrt{n}\sup_{f\in\mathcal{F}}||f(X)||_{\mathbb{P},\infty}\mathbb{E}\big[\sqrt{2\log(N_1(\mathcal{F}',X_{1:n},r/n))}\,\big]). \tag{3.38}$$

Taking $\delta=B_1\sqrt{n}$ and using (3.38) we obtain

$$\mathcal{R}_{\mathrm{ave}}(\Lambda(\mathcal{F}),(X,Y)_{1:n})\le\frac{2-\alpha}{1-\alpha}B_1\sqrt{n}\,(1+\mathbb{E}\big[\sqrt{2\log\left(N_1(\mathcal{F}',X_{1:n},B_1/n)\right)}\,\big]).$$

This inequality, when used to estimate the right-hand side of (3.18), gives the right hand side of (3.36). □

Remark 3.22. As the proof shows, we obtain the same conclusion if $\mathcal{F}$ and $\mathcal{F}'$ are simply assumed to satisfy $\mathcal{R}_{\mathrm{ave}}(\mathcal{F},X_{1:n})\le\mathcal{R}_{\mathrm{ave}}(\mathcal{F}',X_{1:n})$, in particular for $\mathcal{F}\subset(c\,o(\mathcal{F}'))^+$ by a novel application of Talagrand's contraction lemma. Notice also that a slightly bigger upper bound is obtained in place of (3.36) (some terms are multiplied by 2) if we replace (3.36) by the less restrictive condition

$$\mathcal{F}'\subset\mathcal{F}\subset\mathrm{cobal}(\mathcal{F}')$$

3.3.3 A Rademacher confidence interval for the estimator of ES − VaR

In what follows, we will focus on the estimator $\hat{r}_{\hat{q}}$ of $r=s-q$ obtained under the following assumption corresponding to the scheme for approximating r in Algorithm 3.1.

Assumption 3.23. *Assume that $\varphi_\varsigma\equiv\varphi^{(B)}$ in (3.19) (see below) is given by the square loss with truncation on the response*

$$\varphi^{(B)}(y,v,z)=\;\;(z-T_B\left((1-\alpha)^{-1}\,(y-v)^+\right))^2 \tag{3.39}$$

for $B>0$, and that $\mathcal{G}$ is a family of functions $S\to[0,B]$.

As seen in Remark 3.6, the choice (3.39) corresponds to an approximation scheme (with an additional truncation) for the case $g(z)=z^2$. We will also consider the family $\varphi_f^{(B)}(\mathcal{G})$ defined (for f fixed) by

$$\varphi_f^{(B)}(\mathcal{G})=\{(x,y)\mapsto\varphi_f^{(B)}(g)(x,y)=\varphi^{(B)}(y,f(x),g(x))|g\in\mathcal{G}\}.$$

Let us denote by r_f ($f \in \mathcal{L}_+(S)$) any function satisfying

$$r_f(X) = \mathbb{E}[(1-\alpha)^{-1}(Y - f(X))^+ | X], \qquad \mathbb{P}\text{-}a.\,s.,$$

and let $r_f^{(B)}: S \to [0, B]$ be one of its truncated companions, defined by

$$r_f^{(B)}(X) = \mathbb{E}[T_B((1-\alpha)^{-1}(Y - f(X))^+) | X], \qquad \mathbb{P}\text{-}a.\,s.$$

For every $(f, g) \in \mathcal{L}(S) \times \mathcal{L}^+(S)$, we will define

$$h_{(f,g)}(X, Y) = \varphi^{(B)}(Y, f(X), g(X)) - \varphi^{(B)}(Y, f(X), r_f^{(B)}(X)),$$

which is the same as the function in [Barrera, 2022](Section 3, eqn. (4.5)) for the case in consideration.

Lemma 3.24. *For every $(f, f', B) \in \mathcal{L}_+(S) \times \mathcal{L}_+(S) \times (0, \infty]$ and every $p \geq 1$, the inequalities*

$$\begin{aligned} ||r_f - r_f^{(B)}||_{\mathbb{P}_X,p} &\leq ||((1-\alpha)^{-1}(y-f)^+ - B)^+||_{\mathbb{P}_X,p} \\ ||r_f^{(B)} - r_{f'}^{(B)}||_{\mathbb{P}_X,p} &\leq (1-\alpha)^{-1}||f - f'||_{\mathbb{P}_X,p} \end{aligned}$$

hold (with $r_f^{(\infty)} \equiv r_f$).

Proof. The first inequality is a direct consequence of Jensen's inequality:

$$\mathbb{E}[|\mathbb{E}[(W - T_B W)|X]|^p] \leq \mathbb{E}[|W - T_B W|^p] = \mathbb{E}[((|W| - B)^+)^p],$$

valid for $p \geq 1$ and any integrable random variable W. As for the second, notice first that for every $(a, b, B) \in \mathbb{R} \times \mathbb{R} \times [0, \infty]$,

$$|T_B a - T_B b| \leq |a - b|.$$

Combining (3.24) with (3.28) and with Jensen's inequality we get, for every $p \geq 1$:

$$\begin{aligned} ||r_f^{(B)} - r_{f'}^{(B)}||_{\mathbb{P}_X,p}^p &= \mathbb{E}[|\mathbb{E}[T_B((1-\alpha)^{-1}(Y - f(X))^+) - T_B((1-\alpha)^{-1}(Y - f'(X))^+)|X]|^p] \\ &\leq \mathbb{E}[\mathbb{E}[|T_B((1-\alpha)^{-1}(Y - f(X))^+) - T_B((1-\alpha)^{-1}(Y - f'(X))^+)|^p | X]] \\ &\leq (1-\alpha)^{-p}||f - f'||_{\mathbb{P}_X,p}^p. \end{aligned}$$

□

Theorem 3.25. *Under Assumption 3.23, given $f \in \mathcal{L}(S)$ and given $\mathcal{G}' \subset \mathcal{G} \subset co(\mathcal{G}') \subset \mathcal{L}_+(S)$ where $\mathcal{G}$ is a family of functions uniformly bounded by B, the inequality*

$$\begin{aligned} ||\hat{r}_f - r||_{\mathbb{P}_X,2} \leq \inf_{g \in \mathcal{G}} ||g - r||_{\mathbb{P}_X,2} + \frac{2}{1-\alpha}||f - q||_{\mathbb{P}_X,2} + \left\| \left(\frac{1}{1-\alpha}(y-q)^+ - B \right)^+ \right\|_{\mathbb{P}_X,2} \\ + B\left((2/\sqrt{n}) \left(\sqrt{2\log(2/\delta)} + 8\left(1 + \mathbb{E}\left[\sqrt{2\log(N_1(\mathcal{G}', X_{1:n}, B/\sqrt{n}))} \right] \right) \right) \right)^{1/2}. \end{aligned} \tag{3.40}$$

holds for every $\delta \in (0, 1)$, with probability at least $1 - \delta$ (and Remark 3.22 also applies).

Proof.

In this proof, $||\cdot||$ denotes either the $L^2_{\mathbb{P}_X}$ seminorm on $\mathcal{L}_+(S)$ or the $L^2_{\mathbb{P}_{X,Y}}$ seminorm on $\mathcal{L}(S \times \mathbb{R})$, the appropriate choice will be always clear (any other norm will be made explicit).

For $f \in \mathcal{F}$, the triangle inequality and Lemma 3.24 yield

$$\begin{aligned} ||\hat{r}_f - r|| &\leq ||\hat{r}_f - r_f^{(B)}|| + ||r_f^{(B)} - r_q^{(B)}|| + ||r_q^{(B)} - r|| \\ &\leq ||\hat{r}_f - r_f^{(B)}|| + (1-\alpha)^{-1} ||\, f - q|| + ||\, ((1-\alpha)^{-1}(y-q)^+ - B)^+||, \end{aligned} \tag{3.41}$$

Now, if (X', Y') is an independent copy of (X, Y), then by the argument leading to [Barrera, 2022](Section 3, eqn. (4.15))[3.5]

$$\begin{aligned} ||\hat{r}_f - r_f^{(B)}||^2 - \inf_{g \in \mathcal{G}} ||g - r_f^{(B)}||^2 &= \mathbb{E}[\varphi^{(B)}(Y', f(X'), \hat{r}_f(X')) - \varphi^{(B)}(Y', f(X'), r_f(X'))|D] \\ &\leq \sup_{g \in \mathcal{G}} \left\{ \frac{1}{n} \sum_{k=1}^{n} (\mathbb{E}[\varphi^{(B)}(Y, f(X), g(X))] - \varphi^{(B)}(Y_k, f(X_k), g(X_k))) \right\} \\ &\quad + \sup_{g \in \mathcal{G}} \left\{ \frac{1}{n} \sum_{k=1}^{n} (\varphi^{(B)}(Y_k, f(X_k), g(X_k)) - \mathbb{E}[\varphi^{(B)}(Y, f(X), g(X))]) \right\}. \end{aligned}$$

We conclude as in the argument for [Barrera, 2022](eqn. (3.38)) that the inequality

$$\begin{aligned} ||\hat{r}_f - r||^2_{\mathbb{P}_X,2} &\leq \inf_{g \in \mathcal{G}} ||\, g - r_f^{(B)}||^2_{\mathbb{P}_X,2} + \frac{2}{\sqrt{n}} \left(\sup_{g \in \mathcal{G}} ||\varphi_f^{(B)}(g)(X,Y)||_{\mathbb{P},\infty} \} \sqrt{2 \log(2/\delta)} \right) \\ &\quad + \frac{4}{n} \mathcal{R}_{\text{ave}}(\varphi_f^{(B)}(\mathcal{G}), (X,Y)_{1:n}) \end{aligned} \tag{3.42}$$

holds for every $\delta \in (0,1)$ with probability at least $1-\delta$. In virtue again of the triangle inequality and the inequality $\sqrt{a^2+b^2} \leq |a| + |b|$, (3.5) and Lemma 3.24 imply that

$$\begin{aligned} ||\hat{r}_f - r_f^{(B)}|| &\leq \inf_{g \in \mathcal{G}} ||\, g - r|| \\ &\quad + (1-\alpha)^{-1} ||\, f - q|| + ||\, ((1-\alpha)^{-1}(Y - q(X))^+ - B)^+|| \\ &\quad + \left(\frac{2}{\sqrt{n}} \left(\sup_{g \in \mathcal{G}} ||\varphi_f^{(B)}(g)(X,Y)||_{\mathbb{P},\infty} \sqrt{2 \log(\frac{2}{\delta})} + \frac{2}{\sqrt{n}} \mathcal{R}_{\text{ave}}(\varphi_f^{(B)}(\mathcal{G}), (X,Y)_{1:n}) \right) \right)^{1/2}. \end{aligned} \tag{3.43}$$

A combination of (3.40) and (3.42) leads to

$$\begin{aligned} ||\hat{r}_f - r|| &\leq \inf_{g \in \mathcal{G}} ||\, g - r|| \\ &\quad + 2\,(1-\alpha)^{-1} ||\, f - q|| + ||\, ((1-\alpha)^{-1}(y-q)^+ - B)^+|| \\ &\quad + \left(\frac{2}{\sqrt{n}} \left(\sup_{g \in \mathcal{G}} ||\varphi_f^{(B)}(g)(X,Y)||_{\mathbb{P},\infty} \sqrt{2 \log(\frac{2}{\delta})} + \frac{2}{\sqrt{n}} \mathcal{R}_{\text{ave}}(\varphi_f^{(B)}(\mathcal{G}), (X,Y)_{1:n}) \right) \right)^{1/2}. \end{aligned} \tag{3.44}$$

Let us now upper bound the Rademacher complexity. Since the function $u \mapsto u^2$ is Lipschitz on $[0, 2\,B]$ with Lipschitz constant $4\,B$, the Talagrand contraction lemma gives

$$\mathcal{R}_{\text{ave}}(\varphi_f^{(B)}(\mathcal{G}), (X,Y)_{1:n}) \leq 4\,B\,\mathcal{R}_{\text{ave}}(\mathcal{G}, X_{1:n}) = 4\,B\,\mathcal{R}_{\text{ave}}(\mathcal{G}', X_{1:n}). \tag{3.45}$$

An application of [Barrera, 2022](eqn. (3.47)) together with (3.45) gives the inequality

$$\mathcal{R}_{\text{ave}}(\varphi_f^{(B)}(\mathcal{G}), (X,Y)_{1:n}) \leq 4\,B\,(\rho + B\sqrt{n}\,\mathbb{E}\big[\sqrt{2 \log(N_1(\mathcal{G}', X_{1:n}, \rho/n))}\big]) \tag{3.46}$$

for every $\rho > 0$, which in turns implies that (taking $\rho = B\sqrt{n}$)

$$\mathcal{R}_{\text{ave}}(\varphi_f^{(B)}(\mathcal{G}), (X,Y)_{1:n}) \leq 4\,B^2\sqrt{n} \left(1 + \mathbb{E}\Big[\sqrt{2 \log(N_1(\mathcal{G}', X_{1:n}, B/\sqrt{n}))}\Big] \right) \tag{3.47}$$

3.5. Taking $Z'_{1:n} = (X,Y)'_{1:n} \sim (X,Y)$ i.i.d., independent of $(X,Y)_{1:n}$ and $\lambda = 1$.

(3.25) follows from a combination of (3.43), (3.47), and the bound

$$\sup_{g\in\mathcal{G}} ||\varphi_f^{(B)}(g)(X,Y)||_{\mathbb{P},\infty} \leq B^2. \qquad \square$$

3.3.4 VC confidence interval for the estimator of ES − VaR

Theorem 3.26. *Under Assumption 3.23, given $f\in\mathcal{L}_+(S)$ and $\mathcal{G}\subset\mathcal{L}_+(S)$, the inequality*

$$\begin{aligned} ||\hat{r}_f - r||_{\mathbb{P}_X,2} &\leq \sqrt{(6\ \lambda-5)}\ \inf_{g\in\mathcal{G}} ||\ g-r||_{\mathbb{P}_X,2} \\ &+ (1+\sqrt{(6\ \lambda-5)})\,(\frac{1}{1-\alpha}||\ f-q||_{\mathbb{P}_X,2} + ||\,(\frac{1}{1-\alpha}(y-q)^+ - B)^+||_{\mathbb{P}_{X,Y},2}) \\ &+ B\sqrt{\frac{2^7\cdot 3}{(\lambda-1)\,n}\left(\log(42)+\log(1/\delta)+\log\left(\mathbb{E}\left[N_1\left(\mathcal{G},X_{1:n},\frac{B}{24\,n}\right)\right]\right)\right)} \end{aligned} \tag{3.48}$$

holds for every $\delta\in(0,1)$ with probability at least $1-\delta$, provided that

$$1<\lambda\leq 13/12. \tag{3.49}$$

Proof. In this case we depart from the estimate (3.40) and we then estimate $||\hat{r}_f - r_f^{(B)}||$ via [Barrera, 2022](Theorem 4.2) to arrive, by an argument as the one leading to (3.43), at the estimate

$$\begin{aligned} ||\hat{r}_f - r|| \leq\ & \sqrt{(6\ \lambda-5)}\ \inf_{g\in\mathcal{G}} ||\ g-r|| \\ +\ & (1+\sqrt{(6\ \lambda-5)})\,((1-\alpha)^{-1}||\ f-q|| + ||\,((1-\alpha)^{-1}(y-q)^+ - B)^+||) \\ +\ & \left(6\left(\epsilon_n(c,\lambda)\vee(\frac{1}{n\,b(c,\lambda)}(\log a(c,\lambda,\epsilon_n(c,\lambda))+\log(2/\delta)))\right)\right)^{1/2} \end{aligned} \tag{3.50}$$

for $(\gamma,\gamma',\eta')\in(0,\infty)^3$, with probability at least $1-\delta$. Restricting λ to the range (3.49) and using the analysis leading to [Barrera, 2022](eqn. (4.41)), the result follows from (3.50). $\square$

Rademacher vs VC: from "small" to "big" data To give a crude comparison between (3.25) and (3.48), first note that, since $\sqrt{6\,\lambda-5}\approx 1$ (under (3.49)), it is reasonable to limit the discussion to a comparison between the terms in the third line of the inequalities (3.25) and (3.48).

The ratio between these two terms is lower bounded (crudely) by

$$(2^3\cdot 3)^{-1/2}\,((\lambda-1)\,\sqrt{n})^{1/2}(\log(42\,\mathbb{E}[N_1(\mathcal{G},X_{1:n},B/(24\ n))]/\delta))^{-1/2},$$

which shows that (3.25) is worse (bigger) than (3.48) provided that

$$\begin{aligned} \sqrt{n} &\geq \frac{2^3\cdot 3}{\lambda-1}\log(42\,\mathbb{E}[N_1(\mathcal{G},X_{1:n},B/(24\ n))]/\delta) \\ &\geq 2^5\cdot 3^2\,(\log(42)+\log(1/\delta)), \end{aligned} \tag{3.51}$$

where in the last inequality we used the upper bound for λ in (3.49).

The first inequality in (1) is an exact (but crude) criterion on the sample size indicating a domain where (3.48) is preferable to (3.25). The inequality between the first and the third terms in (1) can be understood as an "heuristic" criterion for this preference, indicating in particular the heuristic boundary

$$n\geq(2^5\cdot 3^2\cdot\log(42))^2$$

between "small-medium" and "big" data, where we pass from the Rademacher to the VC regimes.

3.3.5 Multiple-α learning

In this part we are interested in learning $\mathrm{VaR}(Y|X)$ for multiple confidence levels $\alpha \in (0,1)$ using a single empirical error minimization. This can help give insights into the sensitivity of $\mathrm{VaR}(Y|X)$ to the confidence level, or into the full distribution of the law of Y given X (e.g. reflected by an histogram representation).

3.3.5.1 Related literature

The simultaneous learning of conditional quantiles for multiple confidence levels and the problem of quantile crossing, i.e. the violation of the monotonicity with respect to the confidence level, are early addressed in [Takeuchi et al., 2006; Koenker, 2004; He, 1997]. We refer the reader to [Moon et al., 2021] for a review of more recent references. To deal with the quantile crossing problem, two strategies for constraints can be considered. The first strategy is to use spaces of functions which are nondecreasing with respect to the confidence level. Meinshausen and Ridgeway, 2006 introduce quantile regression forests, where the predicted quantile of a new point is based on the empirical percentile of the group (i.e. the terminal leaf of each tree) to which this point belongs and thus the monotonicity of the quantile estimates is satisfied by construction. Regarding neural networks, Hatalis et al., 2017 propose a specific initialization scheme for the weights of the output layer, which does not prevent quantile crossings, but appears to reduce them significantly in their experiments. Cannon, 2018 consider the confidence level as an additional explanatory variable and then explore a network such that the estimate is monotone with a defined covariate (confidence level), imposing the non-crossing. Gasthaus et al., 2019 and Padilla et al., 2020 use a (deep) network with multiple outputs, constrained by design to be positive, which are expected to approximate quantile increments. The latter resembles our *multi-α III* approach in Section 3.4.2.2.2, especially when the increments are constrained to be positive. Under our multi-α III approach, however, we sample the confidence level uniformly on a given interval and we further interpolate linearly with respect to the confidence level in order to have a conditional quantile function that is valid for all quantile levels in the interval.

The second strategy is to consider explicitly the non-crossing constraints during the learning phase of the model in form of either hard constraints (that the model must strictly satisfy) or soft constraints (*i.e.* penalization). Once the non-crossing hard constraints are employed, the model is usually learned using primal-dual optimization algorithms. The latter are applicable in a wide class of models, e.g. support vector regression [Takeuchi et al., 2006; Sangnier et al., 2016] and spline regression [Bondell et al., 2010], but notably not in the case of the family of (deep) neural network, because of the computational cost and the poor scalability of projected gradient descent. Therefore, the non-crossing constraints are more preferably embedded in the training of neural networks via a penalty term [Liu and Wu, 2011; Moon et al., 2021], whose penalization weight then becomes an additional hyperparameter that would need to be tuned. In Section 3.4.2.2.1 we use a similar penalization strategy, where, instead of penalizing discrete increments, we penalize the partial derivative of the network with respect to the confidence level as these give more information about the local behavior around training points and do not add an additional hyperparameter which is the size of the discrete increments in confidence levels.

3.3.5.2 Extension of the bounds to multiple-α learning

The various proofs and bounds presented in this paper for a fixed $\alpha \in [0,1]$ can be extended to the multiple-α learning framework where α is now a random variable supported on $[\underline{\alpha}, \bar{\alpha}] \subset (0,1)$ treated as a covariate alongside X. Denoting by $\mathcal{A}$ the Borel σ-algebra on $[\underline{\alpha}, \bar{\alpha}]$, the table below presents the main changes that need to be done in order to have similar results in this new framework:

Single-α	**Multiple-α**
$D = \{(X_j, Y_j)\}_{j=1}^n$ is an i.i.d sample of (X, Y)	$D = \{(\alpha_j, X_j, Y_j)\}_{j=1}^n$ is an i.i.d sample of (α, X, Y)
$\mathcal{S} \otimes \mathcal{R}$	$\mathcal{A} \otimes \mathcal{S} \otimes \mathcal{R}$
$\Lambda(y, v) = \frac{1}{1-\alpha}(y-v)^+ + v$	$\Lambda(\alpha, y, v) = \frac{1}{1-\alpha}(y-v)^+ + v$
$\tilde{\Lambda}(f) = \mathrm{E}[\Lambda(Y, f(X))]$	$\tilde{\Lambda}(f) = \mathbb{E}[\Lambda(\alpha, Y, f(\alpha, X))]$
$\mathcal{F} \subset \mathcal{L}(S)$	$\mathcal{F} \subset \mathcal{L}([\underline{\alpha}, \bar{\alpha}] \times S)$
$\tilde{q} \in \operatorname{argmin}_{f \in \mathcal{F}} \mathrm{E}[\Lambda(Y, f(X))]$	$\tilde{q} \in \operatorname{argmin}_{f \in \mathcal{F}} \mathbb{E}[\Lambda(\alpha, Y, f(\alpha, X))]$
$\hat{q} \in \operatorname{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{k=1}^n \Lambda(Y_k, f(X_k))$	$\hat{q} \in \operatorname{argmin}_{f \in \mathcal{F}} \frac{1}{n} \sum_{k=1}^n \Lambda(\alpha_k, Y_k, f(\alpha_k, X_k))$
$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \geq \frac{c_{\mathcal{F}^*}}{2(1-\alpha)} \lVert \tilde{q} - q \rVert^2_{\mathbb{P}_X, 2}$	$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \geq \frac{c_{\mathcal{F}^*}}{2} \mathbb{E}[\frac{(\tilde{q}(\alpha, X) - q(\alpha, X))^2}{1-\alpha}] \geq \frac{c_{\mathcal{F}^*}}{2(1-\underline{\alpha})} \lVert \tilde{q} - q \rVert^2_{\mathbb{P}_{(\alpha,X)}, 2}$
$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \frac{C_{\mathcal{F}_0^*}}{2(1-\alpha)} \lVert \tilde{q} - q \rVert^2_{\mathbb{P}_X, 2}$	$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \frac{c_{\mathcal{F}_0^*}}{2} \mathbb{E}[\frac{(\tilde{q}(\alpha, X) - q(\alpha, X))^2}{1-\alpha}] \leq \frac{C_{\mathcal{F}_0^*}}{2(1-\bar{\alpha})} \lVert \tilde{q} - q \rVert^2_{\mathbb{P}_{(\alpha,X)}, 2}$
$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \frac{2-\alpha}{1-\alpha} \inf_{f \in \mathcal{F}} \lVert f - q \rVert^2_{\mathbb{P}_X, 1}$	$\tilde{\Lambda}(\tilde{q}) - \tilde{\Lambda}(q) \leq \inf_{f \in \mathcal{F}} \mathbb{E}[\left(\frac{2-\alpha}{1-\alpha}\right) \lvert f(\alpha, X) - q(\alpha, X) \rvert] \leq \frac{2-\bar{\alpha}}{1-\bar{\alpha}} \inf_{f \in \mathcal{F}} \lVert f - q \rVert^2_{\mathbb{P}_{(\alpha,X)}, 1}$
$\Lambda(\mathcal{F})_{1:n}^{(n)} = \{(X_k, Y_k)_{k \in 1:n} \mapsto (\frac{\Lambda(Y_k, f(X_k))}{n})_{k \in 1:n}, f \in \mathcal{F}\}$	$\Lambda(\mathcal{F})_{1:n}^{(n)} = \{(\alpha_k, X_k, Y_k)_{k \in 1:n} \mapsto (\frac{\Lambda(\alpha_k, Y_k, f(X_k))}{n})_{k \in 1:n}, f \in \mathcal{F}\}$

The implementation of this approach using Neural Networks is discussed in Section 3.4.2.2.

3.4 Learning Using Neural Networks

3.4.1 Error bound of the learning algorithm with one-layer neural networks

We apply the previous developments to the estimation of errors from Algorithm 3.1 when one-hidden-layer neural networks with bounded weights are used to define the hypothesis spaces. We consider the following families of functions:

Definition 3.27. *Let $\sigma : \mathbb{R} \to [0,1]$ be a nondecreasing measurable function that is applied element-wise when supplied with a vector as input and let $(d, M, B) \in \mathbb{N} \times \mathbb{N} \times (0, \infty)$. Denote by $\tilde{\mathcal{F}}(d, B, m, \sigma) \subset \mathcal{L}_{\mathbb{R}^d}$ the family of neural networks on $S = \mathbb{R}^d$ with m (or less) units, one hidden layer, activation function σ and Lasso regularization bound B, defined as follows*

$$\tilde{\mathcal{F}}(d, B, m, \sigma) = \{\mathbb{R}^d \ni x \mapsto c_0 + \sum_{k=1}^m c_k \sigma(a_k \cdot x + b_k) \in \mathbb{R} \mid (a_{1:m}, b_{1:m}) \in (\mathbb{R}^d)^m \times \mathbb{R}^m, c_{0:m} \in \mathbb{R}^{m+1} \text{with } \sum_{k=0}^m |c_k| \leq B\}.$$

It is clear that $\tilde{\mathcal{F}}(d,B,m,\sigma)$ is totally bounded by B. Notice also that for all $m\in\mathbb{N}^\star$

$$\tilde{\mathcal{F}}(d,B,1,\sigma)\subset\tilde{\mathcal{F}}(d,B,m,\sigma)\subset\mathrm{co}(\tilde{\mathcal{F}}(d,B,1,\sigma))=\mathrm{cobal}(\tilde{\mathcal{F}}(d,B,1,\sigma)),$$

where $\mathrm{co}(\cdot)$ and $\mathrm{cobal}(\cdot)$ are defined in (3.31) and (3.31). We have from [Barrera, 2022] for all $0<\rho<\frac{B}{2}$:

$$\log\left(N_1(\tilde{\mathcal{F}}(d,B,m,\sigma),X_{1:n},\rho)\right)\leq((2\,d+5)\,m+1)\,(1+\log(12)+\log(B/\rho)+\log(m+1))$$

This estimate can be combined with Theorem 3.21 to give an error estimate for Algorithm 3.1. In the context of this algorithm, we simplify the notation by writing

$$Y_{h_k}(\omega)=h_k(X(\omega),Y(\omega)),\qquad q(X)_{h_k}(x)=h_k(x,q(x))\qquad(k=1,2),$$
$$r_{h_2}(x)=h_2(x,r(x))$$

where q and $r=s-q$ are defined as in (3.3).

Theorem 3.28. *With the notation of Algorithm 3.1 and in (3.27), and for $\tilde{\mathcal{F}}=\tilde{\mathcal{F}}(d,B_1,m,\sigma)$, if Y_{h_1} satisfies Assumption 3.20, then the inequality*

$$c_{B_1}||\hat{f}-q_{h_1}||^2_{\mathbb{P}_X,2}\leq\left(2(2-\alpha)\inf_{f\in\tilde{\mathcal{F}}}||f-q_{h_1}||_{\mathbb{P}_X,1}\right)\wedge\left(C_{B_1}\inf_{f\in\tilde{\mathcal{F}}}||f-q_{h_1}||^2_{\mathbb{P}_X,2}\right)$$
$$+\frac{4(2-\alpha)}{\sqrt{n}}\left(B_2\sqrt{2\log\left(\frac{2}{\delta}\right)}+2B_1\left(1+\sqrt{2((2d+5)m+1)(1+\log(12(m+1)\sqrt{n}))}\right)\right)$$

holds for every $\delta\in(0,1)$ with probability at least $1-\delta$.

Remark 3.29. The discussion in [Padilla et al., 2020] implies that the rates following from these bounds cannot be improved in general, but as proved in [Chen, 2007](Example 3.2.2), the dimension of the feature space can play a role in a variety of examples.

Analogous reasoning, using this time Theorems 3.25 and 3.26 and the observations in Remark 3.22, lead to the following bound on the error of $\hat{g}$ in Algorithm 3.1:

Theorem 3.30. *With the notation of Algorithm 3.1 and in (3.27), for*[3.6] *$\mathcal{G}=(\tilde{\mathcal{F}}(d,B,m,\sigma))^+$, the inequality*

$$||\hat{g}-r_{h_2}||_{\mathbb{P}_X,2}\leq\sqrt{(6\,\lambda-5)}\inf_{g\in\mathcal{G}}||g-r_{h_2}||_{\mathbb{P}_X,2}$$
$$+(1+\sqrt{(6\,\lambda-5)})((1-\alpha)^{-1}||f-q_{h_2}||_{\mathbb{P}_X,2}+||((1-\alpha)^{-1}(y_{h_2}-q_{h_2})^+-B)^+||_{\mathbb{P}_{X,Y},2})$$
$$+\frac{\sqrt{2}B}{\sqrt[4]{n}}\left(2^4\sqrt{(\log(\frac{2\cdot3\cdot7}{\delta})+\frac{((2d+5)m+1)(1+\log(2^5\cdot3^2(m+1)n))}{(\lambda-1)\sqrt{n}}}\right.$$
$$\left.\wedge\sqrt{\sqrt{2\log(2/\delta)}+2^3\left(1+\sqrt{2(d+3)(1+\log(2^3\cdot3\sqrt{n}))}\right)}\right)$$

holds with probability at least $1-\delta$, for every $1<\lambda\leq13/12$ and every $f\in\mathcal{F}$.

3.6. with $(\mathcal{H})^+=\{(h)^+:h\in\mathcal{H}\}$ for any set of functions $\mathcal{H}$.

More generally, we consider feed-forward neural networks with more than one layer in what follows. We define $\mathcal{F}^{d,o,m,n}$, to be the set of functions of the form $\mathbb{R}^d \ni x \mapsto \zeta_{n+1}^{d,o}(x, W, b) \in \mathbb{R}^o$, where:

$$\begin{aligned}\zeta_0^{d,o}(x, W, b) =& x\\ \zeta_i^{d,o}(x, W, b) =& \sigma\left(W_i\, \zeta_{i-1}^{d,o}(x, W, b) + b_i\right), \forall i \in \{1, \ldots, n\}\\ \zeta_{n+1}^{d,o}(x, W, b) =& W_{n+1}\, \zeta_n^{d,o}(x, W, b) + b_{n+1}\end{aligned}$$

and $W_1 \in \mathbb{R}^{m\times d}, W_2 \ldots, W_n \in \mathbb{R}^{m\times m}, W_{n+1} \in \mathbb{R}^{o\times m}$, $b_1, \ldots, b_n \in \mathbb{R}^m, b_{n+1} \in \mathbb{R}^o$. The function σ is called an activation function. We also choose the Softplus activation function, *i.e.* $\sigma(x) = \log(1 + \exp(x))$.

In what follows, we assume a finite i.i.d sample of (X, Y) given by $D_N := \{(X_i, Y_i)\}_{1\le i\le N}$.

3.4.2 Learning the VaR

3.4.2.1 Single-α learning

In this approach, we fix the confidence level $\alpha \in (\frac{1}{2}, 1)$. The goal then is to find an approximation of $\mathrm{VaR}(Y|X)$, at the confidence level α, as a function of X, represented by a neural network from $\mathcal{F}^{d,1,m,n}$, for given m and n. More precisely, we aim to solve the following optimization problem (cf. (3.3) and (3.73)):

$$\tilde{q}_\alpha \in \operatorname*{argmin}_{q\in\mathcal{F}^{d,1,m,n}} \mathbb{E}[(Y - q(X))^+ + (1-\alpha)\, q(X)] \tag{3.52}$$

or, equivalently, find weights

$$(\tilde{W}^\alpha, \tilde{b}^\alpha) \in \operatorname*{argmin}_{W,b} \mathbb{E}[(Y - \zeta_{n+1}^{d,1}(X, W, b))^+ + (1-\alpha)\, \zeta_{n+1}^{d,1}(X, W, b)]. \tag{3.53}$$

Problem (3.53) is then solved numerically by applying a stochastic gradient descent, or an accelerated version of it, to a finite-sample formulation of the problem (cf. step 3 in Algorithm 3.1):

$$(\hat{W}^\alpha, \hat{b}^\alpha) \in \operatorname*{argmin}_{W,b} \frac{1}{N}\sum_{i=1}^{N} [(Y_i - \zeta_{n+1}^{d,1}(X_i, W, b))^+ + (1-\alpha)\, \zeta_{n+1}^{d,1}(X_i, W, b)]. \tag{3.54}$$

Once (3.54) has been solved numericallly (a procedure to which we will refer to as *training* in what follows), we obtain an approximation of $\mathrm{VaR}(Y|X)$, at the confidence level α, given by $\hat{\varphi}^\alpha(X)$, where

$$\hat{\varphi}^\alpha(x) := \zeta_{n+1}^{d,1}(x, \hat{W}^\alpha, \hat{b}^\alpha)^x \in \mathbb{R}^d.$$

Given that the training is done for a single fixed confidence level α, we refer to this approach as the *single-α learning* (or *single-α* for brevity in the numerics). Under this approach, if one is interested in finding the conditional VaR for another confidence level, one has to repeat the training procedure using the new confidence level in the learning problem (3.54).

3.4.2.2 Multiple-α learning

Learning for a continuum of α's In this approach, we assume that we are interested in conditional VaR estimators for a continuum of confidence levels in $[\underline{\alpha}, \bar{\alpha}]$. We randomize α and assume $\alpha \sim \mathcal{U}([\underline{\alpha}, \bar{\alpha}])$. We then consider a finite i.i.d sample $\alpha_1, \ldots, \alpha_N$ of α, independent of D_N. The finite-sample training problem for this approach can be stated as follows:

$$(\hat{W}^{\underline{\alpha},\bar{\alpha}}, \hat{b}^{\underline{\alpha},\bar{\alpha}}) \in \underset{W,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} (Y_i - \zeta_{n+1}^{d+1,1}([\alpha_i, X_i], W, b))^+ + (1-\alpha_i)\, \zeta_{n+1}^{d+1,1}([\alpha_i, X_i], W, b) \tag{3.55}$$

where $[x, y]$ is the vector obtained by concatenating the vector y to the real x. One can also approximately impose the non-crossing of the quantiles by penalizing the sample average of the negative part of the partial derivative $\frac{\partial}{\partial \alpha} \zeta_{n+1}^{d+1,1}([\alpha, X], W, b)$:

$$\begin{aligned}(\hat{W}^{\underline{\alpha},\bar{\alpha}}, \hat{b}^{\underline{\alpha},\bar{\alpha}}) \in \underset{W,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} & (Y_i - \zeta_{n+1}^{d+1,1}([\alpha_i, X_i], W, b))^+ + (1-\alpha_i)\, \zeta_{n+1}^{d+1,1}([\alpha_i, X_i], W, b) \\ & + \lambda (-\frac{\partial}{\partial \alpha} \zeta_{n+1}^{d+1,1}([\alpha_i, X_i], W, b))^+,\end{aligned}$$

where $\lambda > 0$ determines the strength of the penalization. An approximation for $\mathrm{VaR}(Y|X)$ for any $\alpha \in (\underline{\alpha}, \bar{\alpha})$ is then given by $\zeta_{n+1}^{d+1,1}([\alpha, X], \hat{W}^{\underline{\alpha},\bar{\alpha}}, \hat{b}^{\underline{\alpha},\bar{\alpha}})$. Notice that one can compute the derivative in (3.55) fast in closed-form given our neural network parametrization, as $\frac{\partial}{\partial \alpha} \zeta_{n+1}^{d+1,1}([\alpha, X], W, b) = W_{n+1} \frac{\partial}{\partial \alpha} \zeta_n^{d+1,1}([\alpha, X], W, b)$, where

$$\begin{aligned}\frac{\partial}{\partial \alpha} \zeta_0^{d+1,1}([\alpha, X], W, b) &= [1, 0_d] \text{ and, for } i = 1, \ldots, n, \\ \frac{\partial}{\partial \alpha} \zeta_i^{d+1,1}([\alpha, X], W, b) &= (W_i \frac{\partial}{\partial \alpha} \zeta_{i-1}^{d+1,1}([\alpha, X], W, b)) \odot \sigma'(W_i \zeta_{i-1}^{d+1,1}([\alpha, X], W, b) + b_{i-1}).\end{aligned}$$

Here $\odot$ is an element-wise product and σ' is the derivative of σ (applied element-wise). Given the calculations of $\zeta_{n+1}^{d+1,1}([\alpha, X], W, b)$ and $\frac{\partial}{\partial \alpha} \zeta_{n+1}^{d+1,1}([\alpha, X], W, b)$ share many common sub-expressions, the recursions can be done at the same time, i.e. at each $i \in \{0, \ldots, n+1\}$, compute $\zeta_i^{d+1,1}([\alpha, X], W, b)$ and then reuse the common sub-expressions to compute also $\frac{\partial}{\partial \alpha} \zeta_i^{d+1,1}([\alpha, X], W, b)$. In the numerics, we refer to this approach with *multi-α (I)* if we use a non-zero λ, and *multi-α (II)* otherwise.

Learning for a discrete set of α's Another approach for multiple-α learning would be to target a finite set of confidence levels $\alpha^{(1)}, \ldots, \alpha^{(K)}$ in $[\underline{\alpha}, \bar{\alpha}]$ simultaneously and then yield the estimator corresponding to any confidence level $\alpha \in [\underline{\alpha}, \bar{\alpha}]$ via linear interpolation. More precisely, we solve:

$$(\hat{W}^{1,K}, \hat{b}^{1,K}) \in \underset{W,b}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} [(Y_i - \tilde{\zeta}_{n+1}^{d,K}(\alpha_i, X_i, W, b))^+ + (1-\alpha_i)\, \tilde{\zeta}_{n+1}^{d,1}(\alpha_i, X_i, W, b)] \tag{3.56}$$

where:

$$\tilde{\zeta}_{n+1}^{d,K}(\alpha, x, W, b) := [\zeta_{n+1}^{d,K}(\alpha, x, W, b)]_0 + \sum_{j=1}^{K-1} (\min\{\alpha, \alpha^{(j+1)}\} - \alpha^{(j)}) [\zeta_{n+1}^{d,K}(\alpha, x, W, b)]_j \mathbb{1}_{\alpha \geq \alpha_j} \tag{3.57}$$

with $[x]_j$ denoting the j-th component of vector x. Once the optimization problem solved, an approximation for the VaR of Y given X for any $\alpha \in (\underline{\alpha}, \bar{\alpha})$ is this time given by $\tilde{\zeta}_{n+1}^{d+1,K}(\alpha, X, \hat{W}^{\underline{\alpha},\bar{\alpha}}, \hat{W}^{1,K}, \hat{b}^{1,K})$. Notice that one can impose the monotonicity by design

by adding a positive activation function to each neuron in the output layer of $\zeta_{n+1}^{d+1,K}$, except for the first neuron, e.g. by replacing $[\zeta_{n+1}^{d,K}(\alpha,x,W,b)]_j$ with $\sigma([\zeta_{n+1}^{d,K}(\alpha,x,W,b)]_j)$, for all $j\in 1,\dots,K-1$, in (3.57). However we haven't found doing so to be satisfactory numerically and thus we keep the formulation in (3.57) as is. In the numerics, we refer to this approach as *multi-α (III)*.

3.4.3 Learning the ES using a two-steps approach

As in Section 3.4.2.1, we fix the confidence level $\alpha\in(0,1)$. Our aim is to find an approximation of the $\mathrm{ES}(Y|X)$, at the confidence level α, as a function of X that is represented by a neural network from $\mathcal{F}^{d,1,m,n}$, for given m, n. Assuming a representation, or approximation, q_α of the VaR of Y given X at the confidence level α, which we will call VaR candidate, the goal is to solve the following problem (cf. (3.6)):

$$\tilde{s}_\alpha\in \operatorname*{argmin}_{e\in\mathcal{F}^{d,1,m,n}} \mathbb{E}[(\frac{1}{1-\alpha}(Y-q_\alpha(X))^+ + q_\alpha(X) - s(X))^2]$$

for which we can write a finite-sample version in parameter space as follows (cf. step 4 in Algorithm 3.1):

$$(\hat{W}^\alpha,\hat{b}^\alpha)\in \operatorname*{argmin}_{W,b} \frac{1}{N}\sum_{i=1}^{N}[(\frac{1}{1-\alpha}(Y_i-q_\alpha(X_i))^+ + q_\alpha(X_i) - \zeta_{n+1}^{d,1}(X_i,W,b))^2]. \tag{3.58}$$

Although one could formulate multiple-α learning versions as in Section 3.4.2.2, we have found numerically that it significantly degrades the learning and thus we chose to present only the single-α formulation. However, using a transfer learning trick, one can deduce an ES approximation very quickly using a VaR candidate that is in neural network form. Namely, one can look for an ES approximator using a neural network with the same architecture as the one used for the VaR, set the weights of all hidden layers to those of the VaR network and then freeze them. The training of the ES approximator then falls down to a linear regression to determine the weights of the output layer. We show in Section 3.5 that such a scheme is enough to obtain good approximations, while also being very fast (a fraction of a second in our experiments) if one uses highly optimized linear algebra routines such as the ones implemented by cuBLAS for Nvidia GPUs.

3.4.4 Validating VaR and ES learners without groundtruth values

Assuming one has access to the generative process of the data, as it is the case in most quantitative finance problems, one can estimate distances to the groundtruth VaR and ES without directly computing those, using a simple *twin-simulation* trick.

Proposition 3.31. *Let $\check{q}$ and $\check{s}$ be two Borel functions of x (underlying tentative approximations $\check{q}(X)$ and $\check{s}(X)$ of $\mathrm{VaR}(Y|X)$ and $\mathrm{ES}(Y|X)$ at the confidence level α). Introducing two conditionally independent copies*[3.7] *$Y^{(1)}$ and $Y^{(2)}$ of Y given X and denoting $Y^{(1)}\wedge Y^{(2)}=\min\{Y^{(1)},Y^{(2)}\}$, we have*

$$\|\check{s}(X)-\mathrm{ES}(Y|X)\|_{\mathbb{P},2} = \|\check{s}(X)-\check{q}(X)-\mathbb{E}[\frac{1}{1-\alpha}(Y-\check{q}(X))^+|X]\|_{\mathbb{P},2}+\varepsilon, \tag{3.59}$$

3.7. i.e. for any bounded Borel functions φ and ψ, we have $\mathrm{E}[\varphi(Y^{(1)})\,\psi(Y^{(2)})|X]=\mathrm{E}[\varphi(Y^{(1)})|X]\mathrm{E}[\psi(Y^{(2)})|X]$ and $\mathrm{E}[\varphi(Y^{(1)})|X]=\mathrm{E}[\varphi(Y^{(2)})|X]=\mathrm{E}[\varphi(Y)|X]$.

where

$$\begin{aligned}\|\check{s}(X)-\check{q}(X)-\mathbb{E}[\frac{1}{1-\alpha}(Y-\check{q}(X))^+|X]\|^2_{\mathbb{P},2}&=\|\check{s}(X)-\check{q}(X)\|^2_{\mathbb{P},2}\\&+\frac{1}{(1-\alpha)^2}\mathbb{E}[(Y^{(1)}-\check{q}(X))^+(Y^{(2)}-\check{q}(X))^+]\\&-\frac{2}{1-\alpha}\mathbb{E}[(\check{s}(X)-\check{q}(X))(Y-\check{q}(X))^+]\end{aligned}\tag{3.60}$$

and, assuming $F'_{Y|X}(Y)\geq c$, $\mathbb{P}$*-a.s, for some* $c>0$,

$$\varepsilon\leq\frac{1}{c}(1+\frac{1}{1-\alpha})\,\|\mathbb{P}[Y\geq\check{q}(X)|X]-1+\alpha\|_{\mathbb{P},2},\tag{3.61}$$

where

$$\|\mathbb{P}[Y\geq\check{q}(X)|X]-1+\alpha\|_{\mathbb{P},2}=\sqrt{(1-\alpha)(1-\alpha-2\mathbb{P}(Y>\check{a}(X)))+\mathbb{P}[Y^{(1)}\wedge Y^{(2)}>\check{q}(X)]}.\tag{3.62}$$

Proof. We have

$$\|\mathbb{P}[Y\geq\check{q}(X)|X]-1+\alpha\|_{\mathbb{P},2}=\sqrt{\mathbb{E}[\mathbb{P}[Y\geq\check{q}(X)|X]^2]+(1-\alpha)^2-2(1-\alpha)\mathbb{P}[Y\geq\check{q}(X)|X]},$$

where

$$\mathbb{E}[\mathbb{P}[Y\geq\check{q}(X)|X]^2]=\mathbb{E}[\mathbb{P}[Y^{(1)}\geq\check{q}(X)|X]\,\mathbb{P}[Y^{(2)}\geq\check{q}(X)|X]]=\mathbb{P}[Y^{(1)}\wedge Y^{(2)}\geq\check{q}(X)].$$

Thus (3.62) follows. For the ES, if we assume $F'_{Y|X}(Y)\geq c$, $\mathbb{P}$-a.s, for some $c>0$, then we get (3.59), where ε satisfies (3.61). This follows from an application of the triangular inequality yielding (with $\Lambda(y,v)=\frac{1}{1-\alpha}(y-v)^+ + v$) :

$$\begin{aligned}\|\check{s}(X)-\Lambda(Y,\mathrm{VaR}(Y|X))\|_{\mathbb{P},2}\;\leq\;&\|\check{s}(X)-\Lambda(Y,\check{q}(X))\|_{\mathbb{P},2}\\&+\|\Lambda(Y,\mathrm{VaR}(Y|X))-\Lambda(Y,\check{q}(X))\|_{\mathbb{P},2}\end{aligned}$$

One then uses the $(1+\frac{1}{1-\alpha})$-Lipschitz regularity of Λ with respect to the second argument and the $\frac{1}{c}$-Lipschitz regularity of $F^{-1}_{Y|X}$ to get the result. Using the twin-simulation trick again, we get (3.60). □

The expectations in (3.62) and (3.60) can then be estimated via a simply dedoubled Monte Carlo simulation, as opposed to a nested Monte Carlo that would be required to explicitly attempt to approximate conditional expectations. Moreover the accuracy of the dedoubled Monte-Carlo estimates can be controlled by computing confidence intervals. The distance in (3.62) can be interpreted as a distance *in p-values* between the quantile estimate $\check{q}(X)$ and the true quantile $q(X)$, as opposed to a distance directly between values of conditional quantile estimators. If the approximation $\check{q}$ is sufficiently good, i.e. if this distance is sufficiently small (as compared to $1-\alpha$), then (3.60) can be used as a proxy for $\|\check{s}(X)-\mathrm{ES}(Y|X)\|^2_{\mathbb{P},2}$. Note however that, because of the $\frac{1}{1-\alpha}$ factor in (3.61), the inequality in (3.61) becomes crude when α gets close to 1.

In the case where the dedoubled Monte Carlo estimates for the right-hand-sides in (3.62) and (3.60), after having been confirmed to be accurate by drawing enough samples, are not good enough, one can improve the stochastic gradient descent by changing the optimizer and/or its hyperparameters, in first attempt, and then act on the hypothesis spaces, e.g., in the case of neural networks, try to train with more layers/units or better architectures.

We now test the proposed procedures on a Gaussian toy-example and a dynamic initial margin (DIM) case-study. Any minimization of loss functions over $\mathcal{F}^{d,D,m,n}$ or similar sets of neural networks is done using the Adam algorithm of [Kingma and Ba, 2014] over

the parameters W and b along with mini-batching. In both examples, for the multi-α (I) and multi-α (II) learning approaches, we use the bounds $(\underline{\alpha}, \bar{\alpha}) = (0.85, 1 - 10^{-4})$. For the multi-$\alpha$ (III) approach, we use a uniform interpolation grid $\alpha^{(k)} = 1 - 10^{-3} - k\frac{0.15 - 10^{-3}}{20}$, with $k \in \{0, \dots, 20\}$. All neural networks have 3 hidden layers, and twice their input dimensionality as the number of neurons per hidden layer.

3.5 Conditionally Gaussian Toy Model

Here we take for X a standard multivariate normal vector and we assume that, conditional on X, Y is normally distributed. We denote $\mu(x) = \mathbb{E}[Y|X = x]$ and $\sigma(x) = \sqrt{\mathbb{E}[(Y - \mu(x))^2 | X = x]}$.

Then, denoting by Φ the CDF of the standard normal distribution and by φ its density, we have:

$$\mathrm{VaR}(Y|X) = \mu(X) + \sigma(X)\,\Phi^{-1}(\alpha) \tag{3.63}$$

$$\mathrm{ES}(Y|X) = \mu(X) + \frac{1}{1-\alpha}\,\sigma(X)\,\varphi(\Phi^{-1}(\alpha)) \tag{3.64}$$

These will serve us as ground-truth values. In our toy example, we will assume that for some $d \in \mathbb{N}^\star$, $X \sim \mathcal{N}(0, I_d)$ and $Y|X \sim \mathcal{N}(P_1(X), |P_2(X)|^2)$ where P_1 and P_2 are multivariate polynomials of degree 2, i.e. for some coefficients λ and μ we have for every $x = (x_1, \dots, x_d) \in \mathbb{R}^d$: $P_1(x) = \lambda_0 + \sum_{i=1}^d \lambda_i\, x_i + \sum_{1 \le i < j \le d} \lambda_{i,j}\, x_i\, x_j$ and $P_2(x) = \mu_0 + \sum_{i=1}^d \mu_i\, x_i + \sum_{1 \le i < j \le d} \mu_{i,j}\, x_i\, x_j$.

3.5.1 Results

We use a dimension of $d = 25$ for the state space of X, leading to $1 + d + \frac{d(d+1)}{2} = 351$ monomials in the multivariate polynomials P_1 and P_2. The coefficients λ and μ of those monomials are drawn independently from a standard normal distribution. For this example, we use $2^{19} = 524288$ training points and the same number of testing points for computing the errors. For the Adam algorithm, we used 2000 epochs, mini-batching with batches of size $2^{15} = 32768$ a learning rate $\gamma = 0.01$, and the rest of the parameters kept at their default values as in [Kingma and Ba, 2014].

Tables 3.1, as also 3.5, 3.6 and 3.7 below in the DIM case, suggest that the multi-α approaches are competitive compared to the single-α approach by yielding acceptable errors for confidence levels below 99%, while requiring only one single training, as opposed to the single-α approach which requires one training per target confidence level. For very extreme confidence levels, like 99.9%, the *multi-α (III)* approach outperforms all the other approaches. This can be explained by the fact that, even if the target confidence level is hard to reach given a limited training set, the lower confidence levels in the interpolation grid contribute to inferring the VaR for the target confidence level. Table 3.2 confirms that one can rely on the twin-simulation trick to draw mostly similar conclusions as in Table 3.1, without the need to have access to the goundtruth estimators. Note that we computed upper-bounds of 95% confidence intervals for (3.62), instead of the estimates directly in order to be conservative and take into account the potentially high variance in the indicator functions that need to be simulated in order to estimate (3.62). Table 3.3 demonstrates the effectiveness of the penalization term in the multi-α (I) approach to mitigate the quantiles crossing problem. Table 3.3 also shows that the other multiple-α learning approaches, even without directly penalizing the crossing of the quantiles, behave better than a single-α learning in terms of the crossing of the quantiles.

α	0.999	0.995	0.99
Multi-α (I)	0.151 (0.004)	0.060 (0.002)	**0.039** (0.001)
Multi-α (II)	0.161 (0.004)	0.065 (0.002)	0.042 (0.002)
Multi-α (III)	**0.061** (0.002)	**0.051** (0.002)	0.043 (0.001)
Single-α	0.612 (0.043)	0.062 (0.001)	0.044 (0.001)
α	0.98	0.95	0.9
Multi-α (I)	**0.029** (0.001)	0.023 (0.001)	0.018 (0.001)
Multi-α (II)	0.031 (0.001)	0.024 (0.001)	0.019 (0.001)
Multi-α (III)	0.037 (0.001)	0.029 (0.001)	0.025 (0.001)
Single-α	0.032 (0.001)	**0.021** (0.001)	**0.016** (0.001)

Table 3.1. Mean of RMSE errors of learned conditional VaR estimators against groundtruth values in the Gaussian toy-example across multiple runs (standard deviations of RMSE errors are in parentheses). Errors are normalized by dividing by the standard deviation of the groundtruth VaR.

α	0.999	0.995	0.99
Multi-α (I)	0.00020 (0.000010)	0.00021 (0.000009)	0.00027 (0.000008)
Multi-α (II)	0.00023 (0.000013)	0.00024 (0.000013)	0.00029 (0.000013)
Multi-α (III)	**0.00003** (0.000002)	**0.00008** (0.000003)	**0.00024** (0.000008)
Single-α	0.00008 (0.000003)	0.00020 (0.000007)	0.00035 (0.000008)
α	0.98	0.95	0.9
Multi-α (I)	**0.00046** (0.000009)	**0.00157** (0.000020)	0.00379 (0.000060)
Multi-α (II)	**0.00046** (0.000009)	**0.00157** (0.000030)	0.00398 (0.000086)
Multi-α (III)	0.00057 (0.000015)	0.00171 (0.000030)	0.00428 (0.000066)
Single-α	0.00066 (0.000008)	0.00171 (0.000029)	**0.00343** (0.000069)

Table 3.2. Mean across multiple runs of the upper-bounds (standard deviations of those upper-bounds are in parentheses) of 95% confidence intervals of L^2 *p-value error estimates*, i.e. as defined in (3.62), of learned conditional VaR estimators in the Gaussian toy-example.

E	$q_{0.999}(X) < q_{0.995}(X)$	$q_{0.995}(X) < q_{0.99}(X)$	$q_{0.99}(X) < q_{0.98}(X)$
Multi-α (I)	**0.000004** (0.000001)	**0.000005** (0.000002)	**0.000008** (0.000003)
Multi-α (II)	0.000016 (0.000008)	0.000017 (0.000007)	0.000020 (0.000008)
Multi-α (III)	0.000461 (0.000107)	0.000164 (0.000037)	0.002765 (0.000619)
Single-α	0.111117 (0.003184)	0.251983 (0.006574)	0.213348 (0.005818)
E	$q_{0.98}(X) < q_{0.97}(X)$	$q_{0.97}(X) < q_{0.96}(X)$	$q_{0.96}(X) < q_{0.95}(X)$
Multi-α (I)	**0.000022** (0.000007)	**0.000073** (0.000017)	**0.000367** (0.000059)
Multi-α (II)	0.000032 (0.000008)	0.000080 (0.000012)	0.000405 (0.000096)
Multi-α (III)	0.016378 (0.003258)	0.159370 (0.011163)	0.011956 (0.002695)
Single-α	0.272327 (0.005291)	0.316263 (0.006022)	0.336678 (0.004992)

Table 3.3. Empirical estimates of $\mathbb{P}(E)$, for the events E listed in the first row, for learned conditional VaR estimators in the Gaussian toy-example across multiple runs (standard deviations of the empirical estimators are in parentheses).

For the ES learning in the Gaussian toy-example, for brevity, we denote by *"LR using M VaR"* an ES learning using linear regression only for the output layer, as described in Section 3.4.3, and a VaR learned using the approach M as the candidate VaR. For example, *LR using single-α VaR* refers to the linear regression approach for learning the ES, by using a VaR that is learned with the *single-α* approach as the VaR candidate. To demonstrate the effectiveness of these linear regression approach, we also introduce an ES

that is learned by neural regression, by using a neural network in (3.58), without freezing any weights and using the groundtruth VaR as the VaR candidate. Table 3.4 shows that the linear regression approach outperforms the neural regression, no matter which approach is used for learning the VaR candidate in the context of the ES LR approach. The relative performance of the different linear regression approaches in Table 3.4 is explained by the relative performance of the VaR learning approaches, given that the VaR learning error contributes to the ES learning error.

α	0.999	0.995	0.99
NNR using true VaR	0.408 (0.013)	0.106 (0.002)	0.076 (0.002)
LR using single-α VaR	0.536 (0.037)	0.062 (0.001)	0.045 (0.001)
LR using multi-α (I) VaR	1.900 (0.166)	0.068 (0.004)	**0.037** (0.002)
LR using multi-α (II) VaR	2.382 (0.174)	0.082 (0.006)	0.041 (0.002)
LR using multi-α (III) VaR	**0.126** (0.005)	**0.057** (0.002)	0.050 (0.002)
α	0.98	0.95	0.9
NNR using true VaR	0.054 (0.001)	0.041 (0.001)	0.034 (0.001)
LR using single-α VaR	0.034 (0.001)	**0.025** (0.001)	**0.021** (0.001)
LR using multi-α (I) VaR	**0.031** (0.001)	**0.025** (0.001)	0.022 (0.001)
LR using multi-α (II) VaR	0.032 (0.001)	0.026 (0.001)	0.023 (0.001)
LR using multi-α (III) VaR	0.043 (0.002)	0.036 (0.001)	0.030 (0.001)

Table 3.4. Mean of RMSE errors of learned conditional ES estimators against groundtruth values in the Gaussian toy-example across multiple runs (standard deviations of RMSE errors are in parentheses). Errors are normalized by dividing by the standard deviation of the groundtruth ES.

3.6 Dynamic Initial Margin Case Study

A financial application of the quantile learning framework is the learning of path-wise initial margins in the context of XVA computations (see [Albanese et al., 2021]). Let there be given respectively $\mathbb{R}^d$ valued and real valued stochastic proceesses $(X_t)_{t\geq 0}$ and $(\mathrm{MtM}_t)_{t\geq 0}$, where X_t is Markov and represents the state of the market at time t (e.g. diffused market risk factors) and MtM_t represents the mark-to-market (price) of the portfolio of the bank at time t—cumulative price including the cash flows cumulated up to time t, such that $\mathrm{MtM}_{t+\delta}-\mathrm{MtM}_t$ is $\sigma(X_s, t\leq s\leq t+\delta)$ measurable. We ignore risk-free discounting in the notation (while preserving it in the numerical experiments). The initial margin of the bank at time t at the confidence level α, denoted by IM_t, is defined as:

$$\mathrm{IM}_t := \mathrm{VaR}(\mathrm{MtM}_{t+\delta}-\mathrm{MtM}_t | X_t) \tag{3.65}$$

Hence, having simulated paths of $(X_t)_{t\geq 0}$ and $(\mathrm{MtM}_t)_{t\geq 0}$, one can estimate the initial margin at future time steps $t>0$ using either quantile learning, or a brute-force method involving nested simulations.

3.6.1 Estimating IM_t using a nested Monte Carlo

Alternatively, given $t>0$, we can consider a nested Monte Carlo scheme based on n_{outer} i.i.d samples $(X_t^{(1)}, \mathrm{MtM}_t^{(1)}), \ldots, (X_t^{(n_{\mathrm{outer}})}, \mathrm{MtM}_t^{(n_{\mathrm{outer}})})$ of (X_t, MtM_t) and, for each $i\in\{1,\ldots,n_{\mathrm{outer}}\}$, K i.i.d sub-samples

$$\{\mathrm{MtM}_{t+\delta}^{(i,1),[1]}, \ldots, \mathrm{MtM}_{t+\delta}^{(i,n_{\mathrm{inner}}),[1]}\}, \ldots, \{\mathrm{MtM}_{t+\delta}^{(i,1),[K]}, \ldots, \mathrm{MtM}_{t+\delta}^{(i,n_{\mathrm{inner}}),[K]}\}$$

of $\mathrm{MtM}_{t+\delta}$ conditional on $X_t = X_t^{(i)}$. We can then use these sub-simulations to estimate the conditional quantile in (3.65) for each realization of X_t. For GPU memory limitation reasons, and in order to avoid having to store simulations on the global memory, we chose to do so via one stochastic approximation algorithm per (conditional on) each outer simulation node. More precisely, for every $i \in \{1, \dots, n_{\text{outer}}\}$, we define iteratively over $k \in \{1, \dots, K\}$:

$$\mathrm{IM}_t^{(i),[k+1]} := \mathrm{IM}_t^{(i),[k]} + \gamma \left(p^{(i),[k]} - 1 + \alpha \right) \tag{3.66}$$

where γ is a positive learning rate (see below) and

$$p^{(i),[k]} := \frac{1}{n_{\text{inner}}} \sum_{j=1}^{n_{\text{inner}}} \mathbf{1}_{\{\mathrm{MtM}_{t+\delta}^{(i,j),[k]} - \mathrm{MtM}_t^{(i)} \geq \mathrm{IM}_t^{(i),[k]}\}}.$$

One then iterates over k, simultaneously for all i in parallel, until convergence in order to obtain an approximation of IM_t at each outer realization of X_t. In practice, we take γ to be of the order of the conditional standard deviation of $\mathrm{MtM}_{t+\delta} - \mathrm{MtM}_t$, itself estimated via the same nested Monte Carlo procedure, to speed-up the convergence, and we use Adam instead of plain vanilla stochastic gradient descent. We use $n_{\text{inner}} = 1024$ samples for the sub-simulations and $K = 256$ iterations, enough to achieve an error in p-value, as computed using (3.62), of roughly $0.5\,(1-\alpha)$ in our experiments. In order to accelerate convergence, we use a Gaussian VaR as the initial value (i.e. $\mathrm{IM}_t^{(0),[k]}$), computed using conditional expectation and standard deviation estimates using the inner samples at the first iteration.

3.6.2 Results

We consider a portfolio composed of 100 interest rate swaps with randomly drawn characteristics and final maturity 10 years, assessed in the market model of [Abbas-Turki et al., 2021], with $d = 29$. The *sawtooth*-like behaviour in the paths of $(\mathrm{IM}_t)_{t \geq 0}$ that is visible in the plots in Figure 3.1 is expected, due to the recurring cash-flows inherent to interest rate swaps [Andersen et al., 2017]. We use a multi-factor market model with 10 short-rate processes representing 10 economies and 9 cross-currency rate processes. Given that swap coupons can depend on short-rates at previous fixing dates, we also include in the regression basis the same short-rates but observed at the latest previous fixing date, which leads in total to a dimensionality of 29 for the state vector X_t at a given time $t > 0$, with 100 time steps uniformly spread between time 0 and the final maturity of the portfolio equal to 10 years. Here we use $2^{22} = 4194304$ simulated paths (generated in 25 seconds using the code developed in [Abbas-Turki et al., 2021]) for training and 2^{14} simulated paths, independent of the former, for evaluating the nested Monte Carlo benchmark and computing the errors. We leveraged the transfer learning trick used in [Abbas-Turki et al., 2021], which consists in doing the training starting from the latest time-step and then proceeding backwards by reusing the solution obtained at a time-step t_{k+1} as an initialization for the learning to be done at time t_k. This allows us to use only 16 training epochs. As in the Gaussian toy-example, we use mini-batching. The batch size is taken to be $2^{17} = 131072$ and we use a learning rate of 0.001, and the rest of the Adam parameters are kept at their default values.

To illustrate that the quantile learning approach allows one to learn an entire stochastic process (dynamic initial margin), we plot the mean and 5-th/95-th percentiles of the learned IM process at each time-step for the different quantile learning schemes in Figure 3.1.

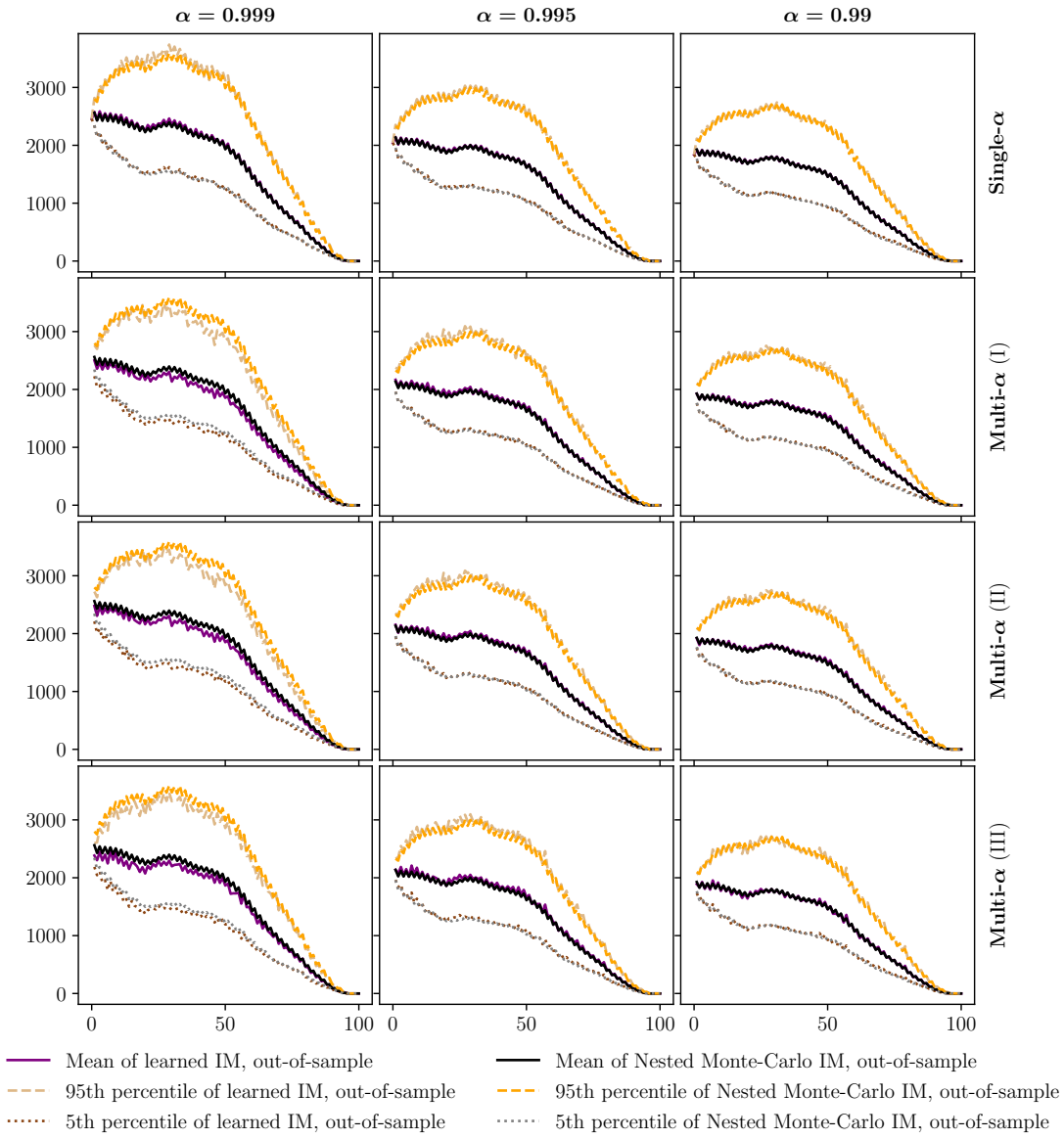

Figure 3.1. Mean and 5-th/95-th percentiles of both the learned and the nested Monte Carlo IM at different time steps and for different values of α and learning approaches. The learning approach used for the plots in each row is indicated on the right, and each column corresponds to one value of α which is indicated at the top of each column. Statistics are computed using out-of-sample trajectories of the diffused risk-factors, and the time steps are on the x-axis.

Tables 3.5, 3.6 and 3.7 (using the nested Monte Carlo as a benchmark, cf.) confirms the conclusions of Table 3.1 regarding the competitiveness of the multi-α approaches.

α	0.999	0.995	0.99	0.98	0.95	0.9
Multi-α (I)	0.265	0.160	0.109	0.065	0.058	**0.056**
Multi-α (II)	0.261	0.155	0.107	0.066	**0.057**	**0.056**
Multi-α (III)	**0.128**	0.185	0.102	0.133	0.116	0.074
Single-α	0.134	**0.074**	**0.070**	**0.056**	0.066	0.065

Table 3.5. RMSE errors of learned IM_t estimators against nested Monte Carlo estimators, for $t = 2.5$years. Errors are normalized by dividing by the standard deviation of the nested Monte Carlo benchmark.

α	0.999	0.995	0.99	0.98	0.95	0.9
Multi-α (I)	0.204	0.166	0.131	0.072	0.061	0.069
Multi-α (II)	0.212	0.162	0.127	0.072	0.062	0.069
Multi-α (III)	**0.150**	0.123	**0.067**	0.065	0.066	0.068
Single-α	0.165	**0.095**	0.070	**0.057**	**0.060**	**0.066**

Table 3.6. RMSE errors of learned IM_t estimators against nested Monte Carlo estimators, for $t = 5$years. Errors are normalized by dividing by the standard deviation of the nested Monte Carlo benchmark.

α	0.999	0.995	0.99	0.98	0.95	0.9
Multi-α (I)	0.292	0.119	0.122	0.095	0.073	0.072
Multi-α (II)	0.296	0.118	0.118	0.091	0.071	0.070
Multi-α (III)	0.157	0.118	0.090	0.089	0.079	0.086
Single-α	**0.119**	**0.088**	**0.082**	**0.068**	**0.061**	**0.061**

Table 3.7. RMSE errors of learned IM_t estimators against nested Monte Carlo estimators, for $t = 7.5$years. Errors are normalized by dividing by the standard deviation of the nested Monte Carlo benchmark.

Conclusion These numerical experiments suggest that learning multiple quantiles (multi-α (I), multi-α (II) or multi-α (III)), although counter-intuitive at first, can help better target extreme quantiles compared to a standard single quantile learning approach. This can be explained by the fact that multiple quantile approaches leverage the information given by nearby quantiles and thus are better at extrapolating at the extremes. The multi-α (II) approach is remarkably good at ensuring, via soft-constraints on the derivative with respect to the quantile level, monotonicity, in cases where consistency among different quantile levels is desired. Our experiments also show that one can successfully use these quantile estimation methods in an XVA or risk calculation setting, where the computation times may be greatly accelerated by replacing nested Monte Carlo estimations by quantile and expected-shortfall learnings.

3.A Value-at-Risk and Expected Shortfall Representations

In this appendix we recall various elicitability results underlying our VaR and ES learning algorithms.

We start by reminding that a cumulative distribution function (c.d.f.) $F\colon \mathbb{R} \to [0,1]$ is by definition integrable if

$$\int_{\mathbb{R}} |y| \; F(dy) < \infty \tag{3.67}$$

where the integration is in the sense of Stieltjes. If Y is a random variable with cumulative distribution function F, then (3.67) holds if and only if Y is $\mathbb{P}$-integrable (the left-hand side of (3.67) is $\mathbb{E}[|Y|]$).

Definition 3.32. *Let $F\colon \mathbb{R} \to [0,1]$ be an integrable c.d.f. and let $\alpha \in (0,1)$. The* value-at-risk *(VaR) and* expected shortfall *(ES) of F at the confidence level α are defined respectively by*

$$\mathrm{VaR}(F) = \inf F^{-1}([\alpha, 1]), \qquad \mathrm{ES}(F) = \frac{1}{1 - F(\mathrm{VaR}(F) -)} \int_{[\mathrm{VaR}(F), \infty)} y \; F(dy). \tag{3.68}$$

where $F(y_0-)=\lim_{y\uparrow y_0}F(y)$. *If* Y *is an integrable random variable on the probability space* $(\Omega,\mathcal{A},\mathbb{P})$, *we write*

$$\mathrm{VaR}(Y)=\mathrm{VaR}(F_Y),\qquad \mathrm{ES}(Y)=\mathrm{ES}(F_Y)$$

where F_Y *is the cumulative distribution function of* Y.

Remark 3.33. If Y is an integrable random variable, then it is easy to see that

$$\mathrm{VaR}(Y)=\inf\{t\colon \mathbb{P}[Y\le t]\ge\alpha\},\qquad \mathrm{ES}(Y)=\mathbb{E}[Y\,|Y\ge\mathrm{VaR}(Y)] \tag{3.69}$$

(the conditional expectation is with respect to $\mathbb{P}$). In particular,

$$\mathrm{VaR}(Y)\le\mathrm{ES}(Y), \tag{3.70}$$

with equality if and only if

$$\mathbb{P}[Y\le\mathrm{VaR}(Y)]=1. \tag{3.71}$$

The versions of (3.69), (3.70) and (3.71) for abstract cumulativbe distribution functions F are clear *mutatis mutandis*.

Remark 3.34. It is necessary to assume that our random variables are bounded (possibly after transformation as explained in Sections 3.2.2 and 3.B) in order to obtain nonasymptotic bounds in the errors induced by the methods to approximate VaR and ES presented here (see for instance (3.18)).

This entails no loss of generality for VaR. To see why, let Y be any integrable random variable defined on $(\Omega,\mathcal{A},\mathbb{P})$, let $I\subset\mathbb{R}$ be a (possibly infinite) interval supporting Y ($\mathbb{P}[Y\in I]=1$), let $-\infty<a<b<\infty$, and let $h\colon I\to(a,b)$ be any increasing bijective, Borel measurable function. Then by monotonicity

$$\mathrm{VaR}(h(Y))=h(\mathrm{VaR}(Y)),$$

which allows us to reduce the approximation of $\mathrm{VaR}(Y)$ to the bounded case: to approximate $\mathrm{VaR}(Y)$, approximate $\mathrm{VaR}(h(Y))$ and compose with h^{-1}. The error bounds provided in this paper, which apply to $\mathrm{VaR}(h(Y))$, can then be translated into error bounds on the approximation of $\mathrm{VaR}(Y)$ using *ad hoc* analytic properties of h.

As for ES, notice that for such h

$$\mathrm{ES}(h(Y))\mathbb{1}_{\{h(Y)\ge\mathrm{VaR}(h(Y))\}}=\mathbb{E}[h(Y)|\mathbb{1}_{\{h(Y)\ge\mathrm{VaR}(h(Y))\}}]=\mathbb{E}[h(Y)|\mathbb{1}_{\{Y\ge\mathrm{VaR}(Y)\}}].$$

From this it follows that if h is in addition convex [concave] on $I\cap[\mathrm{VaR}(F),\infty)$, then[3.8]

$$\mathrm{ES}(Y)\le[\ge]h^{-1}(\mathrm{ES}(h(Y))).$$

The inequality (3.8) for convex [concave] h shows that $h^{-1}(\mathrm{ES}(h(Y)))$ is a *conservative [risky] estimate* of $\mathrm{ES}(Y)$.

Notice that conservative estimates as before are available only when Y is assumed upper bounded, for there is no convex, increasing and bounded bijection with domain $[a,\infty)$. Note also that if h is an increasing affine transformation, then $\mathrm{ES}(h(Y))=h(\mathrm{ES}(Y))$.

3.8. If Z is an integrable random variable on $(\Omega,\mathcal{A},\mathbb{P})$ and $\mathcal{A}_0\subset\mathcal{A}$ is a sigma-algebra, then for every convex, bijective and bimeasurable function $h\colon\mathbb{R}\to\mathbb{R}$,

$$h^{-1}(\mathbb{E}[h(Z)|\mathcal{A}_0])\ge\mathbb{E}[h^{-1}(h(Z))|\mathcal{A}_0]=\mathbb{E}[Z|\mathcal{A}_0].$$

If $\mathcal{A}_0=\sigma(\mathbb{1}_{\{Y\ge a\}})$ and the invertible, bimeasurable function $h\colon\mathbb{R}\to\mathbb{R}$ is convex in the interval $J=I\cap[a,\infty)$ where $\mathbb{P}[Y\in I]=1$, then $h_0=h\mathbb{1}_{I\cap[a,\infty)}+h_1\mathbb{1}_{\mathbb{R}\setminus(I\cap[a,\infty))}$ is convex, invertible and bimeasurable in $\mathbb{R}$ for an appropriate $h_1\colon\mathbb{R}\to\mathbb{R}$, and $\mathbb{E}[h(Z)|\mathcal{A}_0]=\mathbb{E}[h_0(Z)|\mathcal{A}_0]=\mathbb{E}[h_0(Z)|Y\ge a]\mathbb{1}_{\{Y\ge a\}}$. Even more, $\mathbb{E}[h_0(Z)|Y\ge a]=\mathbb{E}[h(Z)|Y\ge a]$ because $h_0|_{I\cap[a,\infty)}=h|_{I\cap[a,\infty)}$. The argument for concave h is similar.

It is convenient for what follows to present the discussion in terms of distribution functions. We start by noticing that if F has an α-quantile, namely if

$$F(y) = \alpha \qquad \textit{for some } y,$$

then $\mathrm{VaR}(F)$ is the minimum of such y's. In this case (and this case only)

$$F(\mathrm{VaR}(F)) = \alpha. \tag{3.72}$$

By the intermediate value theorem, such y exists in $[a,b]$ if

Assumption 3.35. *There exists an interval $[a,b]$ where F is continuous and $F(a)<\alpha\leq F(b)$.*

The following operator will allow us to characterize VaR and ES as minimizers of a suitable functional.

Definition 3.36. *Given a Polish space S, a (Borel measurable) function $h\colon S\times\mathbb{R}\to\mathbb{R}$ and a distribution function F, we define $(h*F)\colon S\to\mathbb{R}$ by*

$$(h*F)(x) = \int_{\mathbb{R}} h(x,y)\; F(dy)$$

provided that $h(x,\cdot)$ is F-integrable for all x. When necessary, we will write $(h*F)(\cdot) = h(\cdot,y)*F(dy)$.

Our methods are built over the following results of [Rockafellar and Uryasev, 2000][3.9], whose easy proof we give for the sake of completeness:

Lemma 3.37. *Let $f\colon\mathbb{R}\to\mathbb{R}$ be an increasing, continuously differentiable function, and let $\Lambda_f\colon\mathbb{R}^2\to\mathbb{R}$ be the tilted loss at level α (given f) defined by*

$$\Lambda_f(y,v) = (1-\alpha)^{-1}(f(y)-f(v))^+ + f(v). \tag{3.73}$$

If F is an integrable distribution function satisfying Assumption 3.35, then the set of minimizers of the function $(\Lambda_f * F)|_{[a,b]}$ is the set of α-quantiles of F within $[a,b]$, and given $c>0$,

$$\mathrm{ES}(F) = \frac{1}{c}\min_v (\Lambda_{c\cdot} * F)|_{[a,b]}(v),$$

Proof. Since f is increasing and continuous, and since F is absolutely continous in $[a,b]$, the identity

$$\begin{aligned}(\Lambda_f * F)'(v) &= \frac{d}{dv}\left((1-\alpha)^{-1}\int_v^\infty (f(y)-f(v))\,F(dy) + f(v)\right)\\ &= f'(v)\,(1-(1-\alpha)^{-1}(1-F(v))).\end{aligned}$$

holds for $v\in[a,b]$. It follows in particular that the (continuously differentiable) function $(\Lambda_f * F)|_{[a,b]}$ has critical points in the set of α-quantiles of F within $[a,b]$. Since F is increasing, these critical points are the minimizers of $\Lambda_f * F$.

With this, (3.73) is a straigthforward consequence of the definition (3.68) of $\mathrm{ES}(F)$ together with (3.72): given any α-quantile q of F within $[a,b]$, and since F is constant in $[\mathrm{VaR}(F),q)$,

$$\begin{aligned}\mathrm{ES}(F) &= \frac{1}{1-\alpha}\int_q^\infty y\,F(dy) = \frac{1}{1-\alpha}\int_{\mathbb{R}}(y-q)^+\,F(dy) + q\\ &= \frac{1}{c}\,(\Lambda_{c\cdot} * F)\,|_{[a,b]}(q) = \frac{1}{c}\min_v (\Lambda_{c\cdot} * F)|_{[a,b]}(v),\end{aligned}$$

3.9. where we only added ι for the sake of data transformation to boundedness.

where for the last equality we used the first part already proved. □

Notice that the estimation of ES via (3.73) implies the estimation of an integral with respect to F. It is desirable, in order to propose distribution-free methods for the estimation of ES, to have characterizations of this risk measure as a *minimizer* (rather than a minimum). The following theorem presents the first one, which works given a corresponding α-quantile:

Lemma 3.38. *Let $\varsigma\colon[0,\infty)\to\mathbb{R}$ be twice continuously differentiable with ς'' positive, and let*

$$\varphi_\varsigma(y,v,z)=\varsigma'(z)\,(z-(1-\alpha)^{-1}\,(y-v)^+)-\varsigma(z) \tag{3.74}$$

be the tilted loss for ES *(given ς). If F is an integrable distribution function and if q is an α-quantile of F, then $\mathrm{ES}(F)-q\in[0,\infty)$ is the unique minimizer of $\varphi_\varsigma(y,q,\cdot)*F\,(dy)|_{[0,\infty)}$.*

Proof. In this case,

$$\frac{d}{dz}\,(\varphi_\varsigma(y,q,z)*F\,(dy))=g''(z)\left(z-(1-\alpha)^{-1}\int_{\mathbb{R}}(y-q)^+\,F\,(dy)\right),$$

whose sign changes at $z{=}\mathrm{ES}(F){-}q$ because $\varsigma''(z){>}0$: this follows as in the proof of (3.73). □

Inspired by Corollary 5.5 in [Fissler and Ziegel, 2016], we finally present the following "joint" function, which is basically a combination of (3.73) and (3.74), for the elicitability (representation as minimizer of an expected loss) of $(\mathrm{VaR},\mathrm{ES})$.

Lemma 3.39. *Let ι,ς be functions $\mathbb{R}\to\mathbb{R}$ where ι' is nonnegative (possibly zero) and continuous, ς' is negative, and ς'' does not vanish, and consider the function $\Lambda_{\iota,\varsigma}\colon\mathbb{R}\times\mathbb{R}\times(-\infty,0]\to\mathbb{R}$ defined by*

$$\begin{aligned}\Lambda_{\iota,\varsigma}(y,v,z)\;&=(1-\alpha)^{-1}\,(\iota(y)-\iota(v))^+-\iota(v)\\&\quad+\varsigma'(z)\,(z-v-(1-\alpha)^{-1}\,(y-v)^+)-\varsigma(z).\end{aligned} \tag{3.75}$$

Then for every integrable c.d.f. F satisfying Assumption 3.35, $(F^{-1}(\alpha)\cap[a,b])\times\{\mathrm{ES}(F)\}$ is the set of minimizers of the function

$$\Lambda_{\iota,\varsigma}(y,\cdot,\cdot)*F\,(dy)\colon[a,b]\times\mathbb{R}\to\mathbb{R}. \tag{3.76}$$

Proof. The derivative of (3.76) with respect to v is

$$(\iota'(v)-\varsigma'(z))\,(1-(1-\alpha)^{-1}\,(1-F(v))) \tag{3.77}$$

which equals zero if and only if $v\in F^{-1}(\alpha)$ by the assumptions on ι' and ς'.

By a similar calculation and using $\varsigma''\neq 0$, the derivative of (3.76) with respect to z is zero if and only if

$$z=v+(1-\alpha)^{-1}\int_{\mathbb{R}}(y-v)^+\,F\,(dy),$$

which, as justified in the proof of Lemma 3.38, gives $z=\mathrm{ES}(F)$ if $v\in F^{-1}(\alpha)$.

It follows that $(F^{-1}(\alpha)\cap[a,b])\times\{\mathrm{ES}(F)\}$ is the set of critical points of (3.76). The fact that these critical points are indeed minimizers of (3.76) follows by an argument akin to the proof of Lemma 3.37 (consider $z=\mathrm{ES}(Y)$ fixed and the expression (3.77) for the derivative with respect to v). □

3.B The Role of Data Transformations and Truncations

The functions $h_k(x,y)$ $(k=1,2)$ in Algorithm 3.1 serve at least two purposes: to uniformly bound and normalize the data, in particular to make it fit to the theory of [Barrera, 2022], and to open the room for profiting from a priori information about the conditional distributions of Y given X.

Let us discuss the functions involved in the estimation of ES: the reason for restricting ourselves to conditionally affine transformations

$$h(x,y)=\tau(x)\,y+\nu(x) \qquad (a(x)>0) \tag{3.78}$$

is that, as explained in Remark 3.34, only these satisfy (in general) the equation

$$\mathrm{ES}(h(X,Y)|X)=h(X,\mathrm{ES}(Y|X)), \tag{3.79}$$

thus allowing us to compute $\mathrm{ES}(Y|X)$ by solving the right hand side of (3.79) for X fixed (which corresponds to the definition of $\hat{r}$ in Algorithm 3.1).

Now, conditionally affine transformations (3.78) are the ones used for "centering and normalizing": typically, one would use $h_2(x,y)=(y-\hat{\mu}(x))/\hat{\sigma}(x)$ where $\hat{\mu}(x)$ and $\hat{\sigma}^2(x)$ are estimates of the conditional mean and variance of Y given that $X=x$.

It may be convenient to say some additional words about this traditional normalization: if $Z\in L^1_{\mathbb{P}}$ has α-quantiles, then integrating the inequality

$$\mathrm{VaR}(Z)\mathbb{1}_{\{Z\geq \mathrm{VaR}(Z)\}}\leq Z\mathbb{1}_{\{Z\geq \mathrm{VaR}(Z)\}} \tag{3.80}$$

and applying Hölder's inequality we obtain the following: for every $p\in[1,\infty]$ $(p'=p/(p-1))$

$$\mathrm{VaR}(Z)\,(1-\alpha)\leq ||\,Z\,||_{\mathbb{P},p}\,(1-\alpha)^{1/p'}.$$

Now, if $F_Z(t):=\mathbb{P}[Z\leq t]$ is continuous and increasing in $[\mathrm{VaR}(Z),\mathrm{VaR}(Z)+\delta)$ (for some $\delta>0$) then

$$-\mathrm{VaR}_\alpha(Z)=\mathrm{VaR}_{(1-\alpha)}(-Z)$$

where $\mathrm{VaR}_\beta(\cdot)$ indicates the corresponding VaR at level β (Definition 3.1), and the previous argument with $-Z$ in place of Z and $1-\alpha$ in place of α leads to

$$-\,\mathrm{VaR}(Z)\,\alpha\leq ||\,Z\,||_{\mathbb{P},p}\,\alpha^{1/p'}. \tag{3.81}$$

Interpreting (3.80), (3.81) in a conditional context and going back to our conventions we obtain that if $p>1$ and $F_{Y|X}$ is continuous and increasing in $[\mathrm{VaR}(Y|X),\mathrm{VaR}(Y|X)+\delta(X))$ then

$$-\,\alpha^{-1/p}\leq\frac{\mathrm{VaR}(Y|X)}{||\,Y\,||_{\mathbb{P}_X,p}}\leq(1-\alpha)^{-1/p} \tag{3.82}$$

which combined with the identity[3.10]

$$\mathrm{ES}(Y|X)=\frac{1}{1-\alpha}\int_\alpha^1\mathrm{VaR}_\beta(Y|X)\;d\,\beta \tag{3.83}$$

gives that

$$-\,p'\,(1-\alpha^{1/p'})\,(1-\alpha)^{-1}\leq\frac{\mathrm{ES}(Y|X)}{||\,Y\,||_{\mathbb{P}_X,p}}\leq p'\,(1-\alpha)^{-1/p}. \tag{3.84}$$

provided that $F_{Y|X}$ is strictly increasing and continuous in $[\mathrm{VaR}(Y|X),\infty)$.

3.10. The equality (3.83) is known as Acerbi's formula. It was generalized to the case of noncontinuous distributions in [Acerbi and Tasche, 2002, Proposition 3.2]. For the case in consideration a quick proof follows by the change of variable $y=F^{-1}_{Y|X}(\beta)=\mathrm{VaR}_\beta(F_{Y|X})$ in (3.2).

The inequalities (3.82) and (3.84) carry at least two important messages: first, the integrability properties of Y are inherited by $\mathrm{VaR}(Y|X)$ and $\mathrm{ES}(Y|X)$; and second, the (conditional) moments of Y control the value of these risk measures. It follows in particular that if $x \mapsto \hat{M}_p(x) > 0$ is (say) an estimate of $x \mapsto M_p(x) := ||Y||_{\mathbb{P}_x,p}$ and $C > 0$ is a constant such that

$$\mathbb{P}[M_p(X) \leq C\,\hat{M}_p(X)] = 1, \tag{3.85}$$

then the specification in Algorithm 3.1 given by

$$h_1(x, y) = h\left(y / \hat{M}_p(x)\right)$$

where $h(y)$ is a continuous and increasing bounded function equal to the identity if

$$|y| \leq C\,(\alpha \wedge (1-\alpha))^{-1/p},$$

permits to assume that

$$B_1 = C\,(\alpha \wedge (1-\alpha))^{-1/p},$$

giving (by the definition of h) that

$$\hat{q}(x) = \hat{M}_p(x)\,\hat{f}(x).$$

As for the computation of $\mathrm{ES} - \mathrm{VaR}$, choosing the conditionally affine transformation

$$h_2(x, y) = y / \hat{M}_p(x)$$

permits to fix the bound

$$C\,(p'\,(1-\alpha)^{-1/p} + \alpha^{-1/p}). \tag{3.86}$$

for the hypotheses $\mathcal{G}$ and to truncate by any B_3 larger than or equal to (3.86) when carrying the regression in Step 4.

Following this line of reasoning, notice that the truncation by B_3 gives rise to a "tail error" of the form

$$\mathbb{E}[((|W| - B_3)^+)^2], \tag{3.87}$$

where $W = (1-\alpha)^{-1}\,(h_2(X,Y) - h_2(X, q(X)(X)))^+$ is the random variable whose conditional expectation (given X) we are trying to estimate.

To justify our belief in the necessity of *a priori* controls on tail bounds on Y (or $Y|X$) for the estimation of ES (e.g. upper bounds to (3.87)), consider the following claim:

Claim: *For every increasing, integrable distribution function F and every $(C, \delta) \in \mathbb{R} \times (0, \infty)$, there exists an increasing and integrable distribution function G coinciding with F in $(-\infty, C]$ and such that $\mathrm{ES}(F) + \delta < \mathrm{ES}(G)$.*[3.11]

According to this claim, no inference can be made in general about $\mathrm{ES}(F)$ only from information on $F(y)$ up to some upper bound $y \leq C < \infty$. Being this is the only kind of information available through finite observations $Y_1(\omega), \ldots, Y_n(\omega)$ of $Y \sim F$, it does not seem possible in general to infer statistical bounds on the approximation error for estimations of $\mathrm{ES}(F)$ which are based only on finite samples of F.[3.12]

3.11. This can be proved easily via the following observation: assume without loss of generality that $\mathrm{VaR}(F) < C$, consider a random variable Y with the distribution F and random variables

$$Y_k = Y \mathbb{1}_{\{Y \leq C\}} + (C + 2^k\,(Y - C))\mathbb{1}_{\{Y > C\}},$$

and notice that $\lim_k \mathrm{ES}(Y_k) = \infty$ by the monotone convergence theorem. The sought for G corresponds to some of these Y_k.

3.12. This is also an obstruction to obtaining in general, from finite samples of (X, Y), a function satisfying (3.85): we have seen that this implies bounds for ES in the case of continuous distributions.

Chapter 4
Pathwise XVAs: The Direct Scheme

This chapter, also submitted as a paper, was co-authored with Lokman Abbas-Turki, Stéphane Crépey and Wissal Sabbagh.

Motivated by the equations of cross valuation adjustments (XVAs) in the realistic case where capital is deemed fungible as a source of funding for variation margin, we introduce a simulation scheme for a class of anticipated BSDEs where the coefficient entails a conditional expected shortfall of the martingale part of the solution. The scheme is explicit in time and uses neural net least-squares and quantile regressions for the embedded conditional expectations and expected shortfall computations. An a posteriori Monte Carlo procedure allows assessing the regression error of the scheme at each time step. The superiority of this scheme with respect to Picard iterations is illustrated in a multi-factor and hybrid market/default risks XVA use-case.

4.1 Introduction

Crépey et al., 2020 showed the existence of a unique solution[4.1] to an anticipated BSDE (ABSDE) in the line of Peng and Yang, 2009, where the coefficient entails a conditional risk measure of a future increment of the martingale part of the solution. Such a coefficient occurs in the equations of cross valuation adjustments (XVAs), accounting for the possibility to use capital at risk as a source of funding for variation margin. In the present paper we address the numerical solution of these equations and its XVA application.

Numerical schemes for BSDEs include backward dynamic programming based on Euler [Bouchard and Touzi, 2004; Zhang, 2004] or higher order [Chassagneux and Crisan, 2014] schemes, combined with regression [Gobet et al., 2005; Huré et al., 2020], Malliavin [Crisan et al., 2010] or cubature [Lyons and Victoir, 2004] methods to estimate the embedded conditional expectations. These admit extensions to jump-diffusions [Bouchard and Élie, 2008], reflected BSDEs [Chassagneux and Richou, 2019], forward-backward SDEs [Delarue and Menozzi, 2006], quadratic BSDEs [Chassagneux and Richou, 2016] or McKean-Vlasov BSDEs [Chassagneux et al., 2019]. Alternative machine learning schemes [E et al., 2017; Teng, 2021] open the door to the numerical solution of BSDEs in dimensions 100 to 1000 (instead of, say, 10 otherwise), however their convergence analysis is still very incomplete. Numerical schemes for BSDEs also include Monte Carlo branching [Henry-Labordere et al., 2017] or Multilevel Picard [Weinan et al., 2019] schemes, but these only provide time 0 estimates (unless they are applied in a nested fashion at each outer node of a nested simulation), making them unsuitable for our (XVA) purpose in this work.

4.1. for square integrable data and solutions.

There is not much literature on the numerical treatment of ABSDEs. As in the present paper (but in a purely Brownian setup), Agarwal et al., 2019 consider an ABSDE involving a conditional expected shortfall as anticipated term (by contrast with a conditional expectation in the previous ABSDE literature). Exploiting the short horizon of the anticipation in the equation (one week in their case versus one year in ours), they devise approximations by standard BSDEs, which allows them to avoid the difficulty posed by the regression of the anticipated terms[4.2]. The XVA ABSDEs received a first numerical treatment in Albanese et al., 2017 on a nested Monte Carlo[4.3] basis, using Picard iteration to decouple the solution from the embedded conditional risk measures and ignoring the conditionings in the latter[4.4] to avoid multiply nested Monte Carlo. The other natural approach to address such problems numerically is regression-based Monte Carlo as introduced above, i.e. iterated regressions (or more general supervised learning algorithms) that are used for cutting the recursively nested levels of Monte Carlo to which a naive implementation of the equations conducts. A first take in this direction, still using Picard iterations for decoupling purposes, was implemented in Albanese et al., 2021, leveraging on the elicitability of the embedded risk measures for learning not only the XVAs, but also these risk measures: conditional value-at-risk for dynamic initial margin calculations and conditional expected shortfall[4.5] for dynamic economic capital calculations. Proceeding in this way allowed Albanese et al., 2021 to learn the embedded conditional risk measures, instead of treating them numerically as constants in Albanese et al., 2017.

In the present paper we introduce an explicit time-discretization scheme which, in conjunction with a refined neural net regression approach for the embedded conditional expectations and expected shortfalls, leads to a direct algorithm for computing the XVA metrics, without Picard iterations. A numerical benchmark of both schemes in a realistic XVA setup emphasizes the superiority of the explicit scheme.

The paper is outlined as follows. Section 4.2 recasts the generic ABSDE of Crépey et al., 2020 in a Markovian setup amenable to numerical simulations. Section 4.3 introduces the related regression-based explicit and implicit/Picard simulation schemes. Section 4.4 provides a numerical benchmark in an XVA setup. Section 4.5 discusses the outputs of the paper in relation with the literature and introduces future research perspectives.

4.1.1 Standing Notation

We denote by:

- $|\cdot|$, an Euclidean norm in the dimension of its arguments;
- $T \in (0, \infty)$, a constant horizon;
- $(\Omega, \mathcal{A}, \mathcal{F}, \mathbb{Q})$, a filtered probability space, for a probability measure $\mathbb{Q}$ on the measurable space $(\Omega, \mathcal{A})$ and a complete and right-continuous filtration $\mathcal{F} = (\mathcal{G}_t)_{0 \le t \le T}$ of sub-σ fields of $\mathcal{A}$;
- $\mathbb{E}$, the $\mathbb{Q}$ expectation, and $\mathbb{P}_t$, $\mathbb{E}_t$, and $\mathbb{ES}_t$, the $(\mathcal{G}_t, \mathbb{Q})$ conditional probability, expectation, and expected shortfall (see (4.1)) at some given quantile level $\alpha \in (\frac{1}{2}, 1)$,

4.2. cf. the beginning of Section 3.2 in Agarwal et al., 2019.

4.3. cf. also Abbas-Turki et al., 2021.

4.4. i.e. computing unconditional risk measures instead of conditional ones.

4.5. that is elicitable jointly with value-at-risk.

where, for each $\mathcal{G}_T$-measurable, $\mathbb{Q}$-integrable, random variable ℓ,

$$\mathbb{ES}_t(\ell) = \mathbb{E}_t(\ell | \ell \geq q_t^\alpha(\ell)), \tag{4.1}$$

in which $q_t^\alpha(\ell)$ denotes the $(\mathcal{G}_t, \mathbb{Q})$ conditional left-quantile of level α[4.6] of ℓ. We recall[4.7] that, for any $\mathcal{G}_T$ measurable, $\mathbb{Q}$ integrable random variables ℓ and ℓ',

$$|\mathbb{ES}_t(\ell) - \mathbb{ES}_t(\ell')| \leq (1-\alpha)^{-1}\mathbb{E}_t[|\ell - \ell'|]. \tag{4.2}$$

4.2 Limiting Equations

We specify the stochastic differential equations adressed from a numerical viewpoint in later sections.

4.2.1 Spaces and Martingale Representation

Given nonnegative integers d and q, we denote by W, an $(\mathcal{F}, \mathbb{P})$ standard d variate Brownian motion, and $\nu = (\nu^k)$, an integer valued random measure[4.8] on $\{0,1\}^q$, with $\mathbb{P}$ compensatrix

$$d\,\mu_t^k = d\,\nu_t^k - \gamma_t^k d\,t, k \in \{0,1\}^q,$$

for some nonnegative real valued predictable process γ^k, $k \in \{0,1\}^q$. Given any positive integer l, we introduce:

- $\mathcal{S}_2^l$, the space of $\mathbb{R}^l$ valued $\mathcal{F}$ adapted càdlàg processes Y such that

$$\|Y\|_{\mathcal{S}_2^l}^2 = \mathbb{E}\left[\sup_{0\leq t\leq T} |Y_t|^2\right] < +\infty;$$

- $\mathcal{H}_2^l$, the space of $\mathbb{R}^{l\otimes d}$ valued $\mathcal{F}$ progressive processes Z such that

$$\|Z\|_{\mathcal{H}_2^l}^2 = \mathbb{E}\left[\int_0^T |Z_t|^2\, d\,t\right] < +\infty;$$

- $\tilde{\mathcal{H}}_2^l$, the space of $\mathbb{R}^{l\otimes 2^q}$ valued $\mathcal{F}$ predictable processes U such that

$$\|U\|_{\tilde{\mathcal{H}}_2^l}^2 = \mathbb{E}\left[\int_0^T |U|_t^2\, d\,t\right] < +\infty, \text{ where } |U|_t^2 = \sum_k \gamma_t^k |U_t^k|^2.$$

In the case where $l = 1$ we drop the index l, e.g. we write $\mathcal{S}_2$ instead of $\mathcal{S}_2^1$.

Assumption 4.1. *Every $(\mathcal{F}, \mathbb{P})$ martingale in $\mathcal{S}_2$ starting from 0 has a representation of the form*[4.9]

$$\int_0^\cdot Z_t\, d\,W_t + \int_0^\cdot U_t\, d\,\mathrm{N}_t, \tag{4.3}$$

4.6. value-at-risk.

4.7. additionally assuming ℓ and ℓ' atomless given $\mathbb{F}_t$, without harm for the XVA applications targeted in this work; cf. e.g. [Barrera et al., 2019](Lemma A.6, Eq. (A.16)).

4.8. see Jacod, 1979.

4.9. using $\int_0^t U_s\, d\,\mathrm{N}_s$ as shorthand for $\sum_{k\in\{0,1\}^q} \int_0^t U_s^k\, d\,\mathrm{N}_s^k$.

for some $Z \in \mathcal{H}_2$ *and* $U \in \tilde{\mathcal{H}}_2$.

4.2.2 The Markovian Anticipated BSDE

Given a positive integer p, let X in $\mathcal{S}_2^p$ satisfy

$$d\,X_t = b(t, X_t)\,d\,t + \sigma(t, X_t)\,d\,W_t, \tag{4.4}$$

for coefficients $b(t,x)$ and $\sigma(t,x)$ Lipschitz in x uniformly in $t \in [0,T]$ and with linear growth in x, so that the SDE (4.4) is classically well-posed in S_2^p, for any constant initial condition $x \in \mathbb{R}^p$. We write $\mathcal{X} = (X, J)$, where a $\{0,1\}^q$ valued "Markov chain like" model component J[4.10] satisfies

$$\mathrm{d}J_t = \sum_{k \in \{0,1\}^q} (k - J_{t-})\,d\,\nu_t^k \tag{4.5}$$

and $\gamma_t = \gamma(t, \mathcal{X}_{t-})$ holds for some continuous functions $\gamma_k(t,x)$ of (t,x), where we write $f(t, \mathcal{X}_t, \cdots)$ as a shorthand for $f_{J_t}(t, X_t, \cdots)$, for any function $f = f_k(t,x,\cdots)$. Hence ν_t^k counts the number of transitions of J to the state k on $(0,t]$.

Given a positive integer l, let $\phi = \phi_k(x)$ define for each k an $\mathbb{R}^l$ valued continuous function on $\mathbb{R}^p$ (and we write $\phi(\mathcal{X}_T)$ as a shorthand for $\phi_{J_T}(X_T)$), $f = f_k(t,x,y,\varrho)$ define for each k an $\mathbb{R}^l$ valued continuous function on $[0,T] \times \mathbb{R}^p \times \mathbb{R}^l \times \mathbb{R}$, and $M \mapsto \mathbb{ES}_\cdot(\Phi_\cdot(M))$ denote a random map from $\mathcal{S}_2^l$ into the space of $\mathcal{F}$ predictable[4.11] processes, where

$$\Phi_{\bar{t}}(M) := \Phi(t; \mathcal{X}_{[t,\bar{t}]}, M_{[t,\bar{t}]} - M_t), \tag{4.6}$$

for some deterministic maps $\bar{t}$ of time t satisfying $\bar{t} \in [t,T]$[4.12] and Φ of time t and càdlàg paths $\mathbf{x}$ and $\mathbf{m}$ on $[t,\bar{t}]$ such that $\mathbf{m}_t$=0. We consider the following anticipated BSDE for Y in $\mathcal{S}_2^l$:

$$Y_t = \mathbb{E}_t\Big[\,\phi(\mathcal{X}_T) + \int_t^T f(s, \mathcal{X}_s, Y_s, \mathbb{ES}_s(\Phi_{s'}(M)))\,d\,s\,\Big], \tag{4.7}$$

where M, also required to belong to $\mathcal{S}_2^l$, is the canonical Doob-Meyer martingale component of the special semimartingale Y.

Assumption 4.2. ***(i)*** *The function* $f = f_k(t,x,y,\varrho)$ *is* Λ_f *Lipschitz in* (y,ϱ) *;*
(ii) *The processes* $\mathbb{ES}_\cdot|\Phi_\cdot(0)|$ *and* $f(\cdot, \mathcal{X}_\cdot, 0, 0)$ *are in* $\mathcal{H}_2$*;*
(iii) Φ *is Lipschitz with respect to its last argument in the sense that for every* $t \in [0,T]$*,*

$$|\Phi(t; \mathbf{x}, \mathbf{m}) - \Phi(t; \mathbf{x}, \mathbf{m}')| \le \Lambda_\Phi\,|\mathbf{m}_{\bar{t}} - \mathbf{m}'_{\bar{t}}| \tag{4.8}$$

holds for all càdlàg paths $\mathbf{x}, \mathbf{m}, \mathbf{m}'$ *on* $[t,\bar{t}]$ *such that* $\mathbf{m}_t = \mathbf{m}'_t = 0$.

Remark 4.3. Assumption 4.2(iii) points out to the case where $\Phi_{\bar{t}}(M)$ only depends on M through $M_{\bar{t}} - M_t$, which indeed corresponds to our XVA use case later below. However, Assumption 4.2(iii) is only a sufficient condition for Lemma 4.5 below to hold, and the algorithms of Section 4.3 are not restricted to this case.

4.10. but with transition probabilities modulated by X.

4.11. assuming the raw process $\Phi_{\bar{t}}(M)$ càdlàg in t, see [Crépey et al., 2020](Lemma 2.1).

4.12. e.g. $\bar{t} = (t+1) \wedge T$ in our XVA use case of Section 4.4.

Lemma 4.4. *There exists a positive constant Λ_ρ such that*

$$|\mathbb{ES}_t(\Phi_{\bar{t}}(M)) - \mathbb{ES}_t(\Phi_{\bar{t}}(M'))|^2 \le \Lambda_\rho^2 \mathbb{E}_t\Big[\int_t^{\bar{t}} (|Z_s - Z'_s|^2 + |U - U'|_s^2)\, d\, s\Big] \tag{4.9}$$

holds for any $M, M' \in \mathcal{S}_2^l$, where (Z, U) and (Z', U') in $\mathcal{H}_2^l \times \tilde{\mathcal{H}}_2^l$ are the integrands in the martingale representations (4.3) of $M - M_0$ and $M' - M'_0$.

Proof. By (4.2), we have

$$\begin{aligned}
&(1-\alpha)^2\, |\mathbb{ES}_t(\Phi_{\bar{t}}(M)) - \mathbb{ES}_t(\Phi_{\bar{t}}(M'))|^2 \le \\
&(\mathbb{E}_t[\Phi(t; \mathcal{X}_{[t,\bar{t}]}, M_{[t,\bar{t}]} - M_t) - \Phi(t; \mathcal{X}_{[t,\bar{t}]}, M'_{[t,\bar{t}]} - M'_t)])^2 \le \\
&\mathbb{E}_t[(\Phi(t; \mathcal{X}_{[t,\bar{t}]}, M_{[t,\bar{t}]} - M_t) - \Phi(t; \mathcal{X}_{[t,\bar{t}]}, M'_{[t,\bar{t}]} - M'_t))^2],
\end{aligned}$$

by the (conditional) Jensen inequality. Moreover, the Lipschitz condition (4.8) yields with $\delta M = M - M'$

$$(\Phi(t; \mathcal{X}_{[t,\bar{t}]}, M_{[t,\bar{t}]} - M_t) - \Phi(t; \mathcal{X}_{[t,\bar{t}]}, M'_{[t,\bar{t}]} - M'_t))^2 \le \Lambda_\Phi^2\, |\delta M_{\bar{t}} - \delta M_t|^2,$$

where, with $\delta Z = Z - Z'$ and $\delta U = U - U'$,

$$\delta M_{\bar{t}} - \delta M_t = \int_t^{\bar{t}} \delta Z_s\, d\, W_s + \int_t^{\bar{t}} \delta U_s\, d\, \mu_s,$$

hence

$$|\delta M_{\bar{t}} - \delta M_t|^2 = \sum_{k=1}^{l} \Big(\int_t^{\bar{t}} \delta Z_s^k\, d\, W_s + \int_t^{\bar{t}} \delta U_s^k\, d\, \mu_s \Big)^2.$$

Therefore,

$$(1-\alpha)^2\, |\mathbb{ES}_t(\Phi_{\bar{t}}(M)) - \mathbb{ES}_t(\Phi_{\bar{t}}(M'))|^2 \le \Lambda_\Phi^2 \mathbb{E}_t \sum_{k=1}^{l} \Big(\int_t^{\bar{t}} \delta Z_s^k\, d\, W_s + \int_t^{\bar{t}} \delta U_s^k\, d\, \mu_s \Big)^2.$$

As a local martingale in $\mathcal{S}_2$, each process $\int_t^{\cdot} \delta Z_s^k\, d\, W_s + \int_t^{\cdot} \delta U_s^k\, d\, \mu_s$ is a square integrable martingale over $[t, \bar{t}]$. The (conditional) Burkholder inequality applied to this process then yields

$$\mathbb{E}_t\Big(\int_t^{\bar{t}} \delta Z_s^k\, d\, W_s + \int_t^{\bar{t}} \delta U_s^k\, d\, \mu_s \Big)^2 \le C \mathbb{E}_t\Big[\int_t^{\bar{t}} (|\delta Z_s^k|^2 + |\delta U^k|_s^2)\, d\, s \Big],$$

so that (4.9) entails (4.9). □

Lemma 4.5. *The ABSDE (4.7) has a unique special semimartingale solution Y in $\mathcal{S}_2^l$ with martingale component M in $\mathcal{S}_2^l$. The process Y is the limit in $\mathcal{S}_2^l$ of the Picard iteration defined by $Y^{(0)} = 0$ and, for $j \ge 1$,*

$$Y_t^{(j)} = \mathbb{E}_t\Big[\phi(\mathcal{X}_T) + \int_t^T f\big(s, \mathcal{X}_s, Y_s^{(j-1)}, \mathbb{ES}_s(\Phi_{\bar{s}}(M^{(j-1)}))\big)\, d\, s \Big], \tag{4.10}$$

where $M^{(j-1)} \in \mathcal{S}_2^l$ is the martingale part of the special semimartingale $Y^{(j-1)} \in \mathcal{S}_2^l$.

Proof. Assumptions 4.2(i) and (ii) imply that the processes

$$\sup_{|y| \le c} |f(\cdot, \mathcal{X}_\cdot, y, \mathbb{ES}_\cdot \Phi_{\bar{\cdot}}(0)) - f(\cdot, \mathcal{X}_\cdot, 0, \mathbb{ES}_\cdot \Phi_{\bar{\cdot}}(0))|^{\frac{1}{2}}$$

(for every $c>0$), as well as $|f(\cdot,\mathcal{X}_\cdot,0,\mathbb{ES}_\cdot(\Phi_\cdot(0))|$, are in $\mathcal{H}_2$, which is [Crépey et al., 2020], whereas [Crépey et al., 2020] are implied by our Assumption 4.2(i) and the Lipschitz property of the functions ϕ_k combined with the standard bound estimate $\|X\|^2_{\mathcal{S}^p_2}\le C\,(1+|x|^2)$ on X (with constant initial condition x). Moreover, (4.9) corresponds to [Crépey et al., 2020]. Hence [Crépey et al., 2020] hold and the result follows by an application of [Crépey et al., 2020](Theorem 3.1). □

4.3 Approximation Schemes

4.3.1 Time Discretizations

Let there be given a deterministic time-grid $0=t_0<t_1<\cdots<t_n=T$ with mesh size[4.13] h. We write $\Delta t_{i+1}=t_{i+1}-t_i$. Let $\bar t_i$ denote an approximation on the grid of $\bar t_i$[4.14]. Let there also be given, on this time grid, simulatable approximations $\mathcal{X}^h$ to $\mathcal{X}$[4.15] and $\Phi^h_{\bar t_i}(M^h)$ to $\Phi_{\bar t}(M)$, with $\Phi^h_{\bar t_i}(M^h)$ of the form[4.16]

$$\Phi^h(t_i;\mathcal{X}^h_{\{t_i,\cdots,\bar t_i\}},M^h_{\{t_i,\cdots,\bar t_i\}}-M^h_{t_i}), \tag{4.11}$$

for some deterministic map Φ^h of grid times t_i and discrete paths $\mathbf{x}^h$ and $\mathbf{m}^h$ on $\{t_i,\cdots,\bar t_i\}$ such that $\mathbf{m}^h_{t_i}$=0.

The explicit time discretization for $(Y,\mathbb{ES}_\cdot(\Phi_\cdot(M)))$ (with M the martingale part of the solution Y to (4.7)) is the process (Y^h,ρ^h) defined at grid times by $Y^h_{t_n}=\phi(\mathcal{X}^h_T)$, $\rho^h_{t_n}=\Phi^h_T(0)$ and, for i decreasing from $n-1$ to 0,

$$\begin{aligned}
&Y^h_{t_i}=\mathbb{E}_{t_i}[Y^h_{t_{i+1}}+f(t_i,\mathcal{X}^h_{t_i},Y^h_{t_{i+1}},\rho^h_{t_{i+1}})\,\Delta t_{i+1}]\\
&\rho^h_{t_i}=\mathbb{ES}_{t_i}\Phi^h_{\bar t_i}\left(Y^h_{t_l}+\sum_{k<l}f(t_k,\mathcal{X}^h_{t_k},Y^h_{t_{k+1}},\rho^h_{t_{k+1}})\,\Delta t_{k+1},l=0,\dots,n\right).
\end{aligned}\tag{4.12}$$

The Picard iteration for the implicit time discretization for $(Y,\mathbb{ES}_\cdot(\Phi_\cdot(M)))$ is defined by the sequence of discrete time processes $(Y^{0,\pi},\rho^{0,\pi})=(0,0)$ and, for each j increasing from 1 to ∞: $Y^{j,h}_{t_n}=\phi(\mathcal{X}^h_T)$, $\rho^{j,h}_{t_n}=\Phi^h_T(0)$ and, for i decreasing from $n-1$ to 0,

$$\begin{aligned}
&Y^{j,h}_{t_i}=\mathbb{E}_{t_i}[Y^{j,h}_{t_{i+1}}+f(t_i,\mathcal{X}^h_{t_i},Y^{j-1,h}_{t_i},\rho^{j-1,h}_{t_i})\,\Delta t_{i+1}],\\
&\rho^{j,h}_{t_i}=\mathbb{ES}_{t_i}\Phi^h_{\bar t_i}\left(Y^{j,h}_{t_l}+\sum_{k<l}f(t_k,\mathcal{X}^h_{t_k},Y^{j-1,h}_{t_k},\rho^{j-1,h}_{t_k})\,\Delta t_{k+1},l=0,\dots,n\right).
\end{aligned}\tag{4.13}$$

The time-consistency of these schemes, i.e. the convergence of the Y^h (resp. $Y^{h,j}$) to Y as h goes to 0 (resp. h goes to 0 and j goes to infinity) will be studied elsewhere. Our focus hereafter is the discretization in space of (4.12), (4.13).

4.13. maximum time step.

4.14. cf. after (4.6).

4.15. e.g. the Euler scheme for X and a related approximation for J.

4.16. cf. (4.6).

4.3.2 Fully Discrete Algorithms

Whenever a process M^h on the time grid is such that

$$M^h_{\{t=t_i,\cdots,\bar{t}_i\}} - M^h_{t_i} \text{ is a measurable functional of } (t_i, \mathcal{X}^h_{t_i}), \ldots, (\bar{t}_i, \mathcal{X}^h_{\bar{t}_i}) \tag{4.14}$$

in view of [Barrera et al., 2022](Theorem 2.3)[4.17], we have (with $\varphi = \varphi_k(t,x)$ and $\phi = \phi_k(t, x)$ below): $\mathbb{ES}_t(\Phi^h_t(M^h)) = (1-\alpha)^{-1}\,\phi^*(t, \mathcal{X}^h_t)$, where

$$\phi^*_\cdot(t,\cdot) = \underset{\phi_\cdot(t,\cdot)\in\mathcal{B}}{\operatorname{argmin}} \mathbb{E}[(\Phi^h_t(M^h)\,\mathbb{1}_{\{\Phi^h_t(M^h)\geq\varphi^*(t,\mathcal{X}^h_t)\}} - \phi(t,\mathcal{X}^h_t))^2], \tag{4.15}$$

in which

$$\varphi^*_\cdot(t,\cdot) = \underset{\phi_\cdot(t,\cdot)\in\mathcal{B}}{\operatorname{argmin}} \mathbb{E}[(\varphi(t,\mathcal{X}^h_t) + (1-\alpha)^{-1}\,(\Phi^h_t(M^h) - \varphi(t,\mathcal{X}^h_t))^+], \tag{4.16}$$

both minimizations bearing over the set $\mathcal{B}$ of the Borel functions of (x,k).

As estimates of $\varphi^*(t,\mathcal{X}^h_t)$ and $\phi^*(t,\mathcal{X}^h_t)$ we use the functions $\hat{\varphi}^*(t,\cdot)$ and $\hat{\phi}^*(t,\cdot)$ obtained by solving the respective problems (4.16) and (4.15) with $\mathcal{B}$ replaced by a to-be-specified hypothesis space of functions, $\mathbb{E}$ by the sample mean over a sufficiently large number of independent realizations of $\mathcal{X}^h$, minimization by (approximate) numerical minimization through Adam stochastic gradient descent [Kingma and Ba, 2014], and φ^* in (4.15) by $\hat{\varphi}^*$.

The fully (time and space) discrete counterparts of (4.12) and (4.13) follow by estimation of the embedded conditional expectations (resp. expected shortfalls) through nonparametric least-squares (resp. quantile as explained above) regressions of suitable features, which we write: $\hat{Y}^h_{t_n} = \phi(\mathcal{X}^h_T)$, $\hat{\rho}^h_{t_n} = \Phi^h_T(0)$ and, for i decreasing from $n-1$ to 0,

$$\begin{aligned}
\hat{Y}^h_{t_i} &= \hat{\mathbb{E}}_{t_i}[\hat{Y}^h_{t_{i+1}} + f(t_i, \mathcal{X}^h_{t_i}, \hat{Y}^h_{t_{i+1}}, \hat{\rho}^h_{t_{i+1}})\,\Delta t_{i+1}] \\
\hat{\rho}^h_{t_i} &= \widehat{\mathbb{ES}}_{t_i}\Phi^h_{\bar{t}_i}\left(\hat{Y}^h_{t_l} + \sum_{k<l} f(t_k, \mathcal{X}^h_{t_k}, \hat{Y}^h_{t_{k+1}}, \hat{\rho}^h_{t_{k+1}})\,\Delta t_{k+1}, l = 0,\ldots,n\right),
\end{aligned} \tag{4.17}$$

respectively $\hat{Y}^{j,h}_{t_n} = \phi(\mathcal{X}^h_T)$, $\hat{\rho}^{j,h}_{t_n} = \Phi^h_T(0)$ and, for i decreasing from $n-1$ to 0,

$$\begin{aligned}
\hat{Y}^{j,h}_{t_i} &= \hat{\mathbb{E}}_{t_i}[\hat{Y}^{j,h}_{t_{i+1}} + f(t_i, \mathcal{X}^h_{t_i}, \hat{Y}^{j-1,h}_{t_i}, \hat{\rho}^{j-1,h}_{t_i})\,\Delta t_{i+1}], \\
\hat{\rho}^{j,h}_{t_i} &= \widehat{\mathbb{ES}}_{t_i}\Phi^h_{\bar{t}_i}\left(\hat{Y}^{j,h}_{t_l} + \sum_{k<l} f(t_k, \mathcal{X}^h_{t_k}, \hat{Y}^{j-1,h}_{t_k}, \hat{\rho}^{j-1,h}_{t_k})\,\Delta t_{k+1}, l = 0,\ldots,n\right).
\end{aligned} \tag{4.18}$$

As in [Abbas-Turki et al., 2021], given weight matrices $A^{[L+1]} \in \mathbb{R}^{1\times u}, \ldots, A^{[\ell]} \in \mathbb{R}^{u\times u}, \ldots, A^{[1]} \in \mathbb{R}^{u\times(p+q)}$, biases $b^{[L+1]} \in \mathbb{R}, \ldots, b^{[\ell]} \in \mathbb{R}^u, \ldots, b^{[1]} \in \mathbb{R}^u$, and a scalar nonlinearity ς applied element-wise, let, for every $z \in \mathbb{R}^{p+q}$,

$$\begin{aligned}
\zeta^{[0]}(z; A, b) &= z \\
\zeta^{[\ell]}(z; A, b) &= \varsigma\,(A^{[\ell]}\zeta^{[\ell-1]}(z; A, b) + b^{[\ell]}), \quad \ell = 1,\ldots,L \\
\zeta^{[L+1]}(z; A, b) &= A^{[L+1]}\zeta^{(L)}(z; A, b) + b^{[L+1]},
\end{aligned}$$

where A and b denote the respective concatenations of the $A^{[\ell]}$ and of the $b^{[\ell]}$. The function $\mathbb{R}^{p+q} \ni z \mapsto \zeta^{[L+1]}(z; A, b) \in \mathbb{R}$ then implements a neural network with L hidden layers, u neurons per hidden layer, ς as an activation function applied on each hidden unit, and no nonlinearity at the output layer. We can then introduce Algorithms 4.2 (for the

4.17. additionally assuming $\Phi^h_t(M^h)$ atomless given $\mathbb{F}_t$.

explicit scheme) and 4.3 (for the implicit/Picard scheme), both using Algorithm 4.1 as an elementary learning block. At the beginning of the algorithms, *i.e.* before any learning is performed, weights are initialized randomly according to classic weight initialization schemes ([Goodfellow et al., 2016]). In our numerical experiments, we use Softplus activation functions in the hidden layers, in combination with the related weight initialization scheme by [He et al., 2015].

Remark 4.6. Assuming the same number of epochs during each run of the elementary learning block of Algorithm 4.1 for all schemes, the Picard scheme with j Picard iterations implies j times more regressions than the explicit scheme. Hence, by design, it is at a computational disadvantage compared to the explicit scheme of Algorithm 4.2. In order to try and make the Picard scheme more competitive, we propose to modify Algorithm 4.3 by leveraging its iterative nature through Picard iterations. Namely, given a computational budget equivalent to n_{train} training epochs in the explicit scheme, assuming that we have a target of j Picard iterations in the implicit scheme, we only train for only n_{train}/j epochs at each Picard iteration, reusing the obtained neural network weights as an initialization for the neural networks at the next Picard iteration, versus the next time step in Algorithms 4.2 and 4.3. This ensures that the total computational cost of j Picard iterations is roughly the same as the cost of learning in the explicit scheme.

This modification is achieved by changing the $i+1$ in lines 12, 14 and 18 of Algorithm 4.3 to I, where $I = i + 1_{\{\text{picard\_iter}=1\}}$, then ensuring that the weights of the previous Picard iteration, as opposed to the previous time step of the same Picard iteration in Algorithm 4.3, are used to initialize each learning.

Algorithm 4.1
Elementary learning block

name: BaseAlg
input: $\{(X^\iota, Y^\iota), \iota \in \mathcal{I}\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E \in \mathbb{N}^\star$, a learning rate $\eta > 0$, initial values for the network parameters A and b, type of regression *regr*
output: Trained parameters A and b

1 **define**

$$\mathcal{L}(A, b, \text{batch}) =$$

2
$$\begin{cases} \dfrac{1}{|\text{batch}|} \displaystyle\sum_{\iota \in \text{batch}} (\zeta^{[L+1]}(X^\iota; A, b) - Y^\iota)^2 & \text{if } regr = \mathcal{L}^2 \\ \dfrac{1}{|\text{batch}|} \displaystyle\sum_{\iota \in \text{batch}} (Y^\iota - \zeta^{[L+1]}(X^\iota; A, b))^+ + (1-\alpha)\, \zeta^{[L+1]}(X^\iota; A, b) & \text{if } regr = \text{quantile} \end{cases}$$

3

4 **for** epoch $= 1 \ldots E$ **do**
5 `// loop over epochs`
6 **for** batch $\in \mathcal{B}$ **do**
7 `// loop over batches`
8 **for** $\ell = 1 \ldots L+1$ **do**
9 $A^{[\ell]} \leftarrow A^{[\ell]} - \eta \nabla_{A^{[\ell]}} \mathcal{L}(A, b, \text{batch})$
10 $b^{[\ell]} \leftarrow b^{[\ell]} - \eta \nabla_{b^{[\ell]}} \mathcal{L}(A, b, \text{batch})$
11 **end**
12 **end**
13 **end**

Algorithm 4.2
Explicit backward learning scheme

name: ExplicitBackwardAlg
input: $\{\{\mathcal{X}_{t_i}^{h,\iota}, 1 \le i \le n\}, \phi(\mathcal{X}_T^{h,\iota}), \iota \in \mathcal{I}\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E \in \mathbb{N}^\star$, a learning rate $\eta > 0$
output: $\hat{\varphi}_1, \dots, \hat{\varphi}_n$

For all $\iota \in \mathcal{I}$, let $y^\iota, M^\iota \in \mathbb{R}^l$
Initialize parameters $\{(A_{n+1,k}, b_{n+1,k}), k \in \{1, \dots, l\}\}$ of the networks, indexed by $k \in \{1, \dots, l\}$, at terminal time-step n
Initialize parameters $\{(A_{n,\mathbb{V}a\mathbb{R}}, b_{n,\mathbb{V}a\mathbb{R}}), (A_{n,\mathbb{ES}}, b_{n,\mathbb{ES}})\}$ of the networks approximating the $\mathbb{V}a\mathbb{R}$ and $\mathbb{ES}$ at terminal time-step n
for $k = 1 \dots l$ **do**
    `// can be skipped if one already has the terminal values in functional form`
    $A_{n,k}, b_{n,k} \leftarrow \text{BaseAlg}(\{(\mathcal{X}_{t_n}^{h,\iota}, \phi(\mathcal{X}_T)^{h,\iota}), \iota \in \mathcal{I}\}, \mathcal{B}, E, \eta, A_{n+1,k}, b_{n+1,k}, \mathcal{L}^2)$
end
Let $y^\iota \in \mathbb{R}^l$ such that its k-th component is $\zeta^{[L+1]}(\mathcal{X}_{t_n}^{h,\iota}; A_{n,k}, b_{n,k})$ for all $k = 1 \dots l$
$M_n^\iota \leftarrow 0$
for $i = n-1 \dots 1$ **do**
    `// Learn the` $\mathbb{V}a\mathbb{R}$ `and the` $\mathbb{ES}$ `in two-steps`
    `// The` $\mathbb{V}a\mathbb{R}$ `is first learned using a quantile regression`
    foreach $\iota \in \mathcal{I}$ **do** $\chi^\iota \leftarrow \Phi(t_{i+1}, (\mathcal{X}_{t_j}^{h,\iota})_{j \in \{i+1, \dots, n\}}, (M_j^\iota)_{j \in \{i+1, \dots, n\}})$
    $A_{i,\mathbb{V}a\mathbb{R}}, b_{i,\mathbb{V}a\mathbb{R}} \leftarrow \text{BaseAlg}(\{(\mathcal{X}_{t_i}^{h,\iota}, \chi^\iota), \iota \in \mathcal{I}\}, \mathcal{B}, E, \eta, A_{i+1,\mathbb{V}a\mathbb{R}}, b_{i+1,\mathbb{V}a\mathbb{R}}, \text{quantile})$
    `// The` $\mathbb{ES}$ `is then deduced using an` $\mathcal{L}^2$ `regression`
    $\chi^\iota \leftarrow \frac{1}{1-\alpha} \chi^\iota 1_{\{\chi^\iota \ge \zeta^{[L+1]}(\mathcal{X}_{t_i}^{h,\iota}; A_{i,\mathbb{V}a\mathbb{R}}, b_{i,\mathbb{V}a\mathbb{R}})\}}$
    $A_{i,\mathbb{ES}}, b_{i,\mathbb{ES}} \leftarrow \text{BaseAlg}(\{(\mathcal{X}_{t_i}^{h,\iota}, \chi^\iota), \iota \in \mathcal{I}\}, \mathcal{B}, E, \eta, A_{i+1,\mathbb{ES}}, b_{i+1,\mathbb{ES}}, \mathcal{L}^2)$
    `// We compute the integrand` χ `that needs to be projected to get the solution of the BSDE at the current time step`
    $\varrho^\iota \leftarrow \zeta^{[L+1]}(\mathcal{X}_{t_{i+1}}^{h,\iota}; A_{i+1,\mathbb{ES}}, b_{i+1,\mathbb{ES}})$
    $\chi^\iota \leftarrow y^\iota + f(t_i, \mathcal{X}_{t_i}^{h,\iota}, y^\iota, \varrho^\iota)\, \Delta t_{i+1}$
    for $k = 1 \dots l$ **do**
        $A_{i,k}, b_{i,k} \leftarrow \text{BaseAlg}(\{(\mathcal{X}_{t_i}^{h,\iota}, [\chi^\iota]_k), \iota \in \mathcal{I}\}, \mathcal{B}, E, \eta, A_{i+1,k}, b_{i+1,k}, \mathcal{L}^2)$
    end
    `// Update` M `to have` $M_i = Y_{t_n}^h - Y_{t_i}^h + \sum_{k=i}^{n-1} f(t_k, \mathcal{X}_{t_k}^h, Y_{t_{k+1}}^h, \rho_{t_{k+1}}^h)\, \Delta t_{k+1}$
    for $\iota \in \mathcal{I}$ **do**
        for $k = 1 \dots l$ **do**
            $[y^\iota]_k \leftarrow \zeta^{[L+1]}(\mathcal{X}_{t_i}^{h,\iota}; A_{i,k}, b_{i,k})$
        end
        $M_i^\iota \leftarrow M_{i+1}^\iota + \chi^\iota - y^\iota$
    end
end

Algorithm 4.3
Picard backward learning scheme

name: PicardBackwardAlg
input: $\{\{\mathcal{X}^{h,\iota}_{t_i}, 1\le i\le n\}, \phi(\mathcal{X}^{h,\iota}_T), \iota\in\mathcal{I}\}$, a partition $\mathcal{B}$ of $\mathcal{I}$, a number of epochs $E\in\mathbb{N}^\star$, a learning rate $\eta>0$ and number picard_iters of picard iterations
output: $\hat{\varphi}_1,\dots,\hat{\varphi}_n$

For all $\iota\in\mathcal{I}$, let $y^\iota, M^\iota\in\mathbb{R}^l$
Initialize parameters $\{(A_{n+1,k}, b_{n+1,k}), k\in\{1,\dots,l\}\}$ of the networks, indexed by $k\in\{1,\dots,l\}$, at terminal time-step n
Initialize parameters $\{(A_{n,\mathbb{V}a\mathbb{R}}, b_{n,\mathbb{V}a\mathbb{R}}), (A_{n,\mathbb{ES}}, b_{n,\mathbb{ES}})\}$ of the networks approximating the $\mathbb{V}a\mathbb{R}$ and $\mathbb{ES}$ at terminal time-step n
// The following can be skipped if one already has the terminal values in functional form
foreach $k=1...l$ **do** $A_{n,k}, b_{n,k} \leftarrow \text{BaseAlg}(\{(\mathcal{X}^{h,\iota}_{t_n}, \phi(\mathcal{X}_T)^{h,\iota}), \iota\in\mathcal{I}\}, \mathcal{B}, E, \eta, A_{n+1,k}, b_{n+1,k}, \mathcal{L}^2)$
foreach $i=n-1\dots1$ **and** $i\in\mathcal{I}$ **do** $M^\iota_{i,\text{prev}}\leftarrow 0$
for picard_iter $=1\dots$picard_iters **do**
  foreach $\iota\in\mathcal{I}$ **do** $M^\iota_{n,\text{current}}\leftarrow 0$
  for $i=n-1\dots1$ **do**
    // Learn the $\mathbb{V}a\mathbb{R}$ and the $\mathbb{ES}$ in two-steps
    // The $\mathbb{V}a\mathbb{R}$ is first learned using a quantile regression
    foreach $\iota\in\mathcal{I}$ **do** $\chi^\iota\leftarrow\Phi(t_i, (\mathcal{X}^\iota_{t_j})_{j\in\{i,\dots,n\}}, (M^\iota_{j,\text{prev}})_{j\in\{i,\dots,n\}})$
    $A_{i,\mathbb{V}a\mathbb{R}}, b_{i,\mathbb{V}a\mathbb{R}} \leftarrow \text{BaseAlg}(\{(\mathcal{X}^{h,\iota}_{t_i}, \chi^\iota), \iota\in\mathcal{I}\}, \mathcal{B}, E, \eta, A_{i+1,\mathbb{V}a\mathbb{R}}, b_{i+1,\mathbb{V}a\mathbb{R}}, \text{quantile})$
    // The $\mathbb{ES}$ is then deduced using an $\mathcal{L}^2$ regression
    $\chi^\iota\leftarrow\frac{1}{1-\alpha}\chi^\iota 1_{\{\chi^\iota\ge\zeta^{[L+1]}(\mathcal{X}^{h,\iota}_{t_i}; A_{i,\mathbb{V}a\mathbb{R}}, b_{i,\mathbb{V}a\mathbb{R}})\}}$
    $A_{i,\mathbb{ES}}, b_{i,\mathbb{ES}} \leftarrow \text{BaseAlg}(\{(\mathcal{X}^{h,\iota}_{t_i}, \chi^\iota), \iota\in\mathcal{I}\}, \mathcal{B}, E, \eta, A_{i+1,\mathbb{ES}}, b_{I,\mathbb{ES}}, \mathcal{L}^2)$
    // We compute the integrand χ that needs to be projected to get the solution of the BSDE at the current time step
    for $\iota\in\mathcal{I}$ **do**
      foreach $k=1\dots l$ **do** $[y^\iota]_k\leftarrow\zeta^{[L+1]}(\mathcal{X}^{h,\iota}_{t_{i+1}}; A_{i+1,k}, b_{i+1,k})$
      $\chi^\iota\leftarrow y^\iota$
      if picard_iter >1 **then**
        foreach $k=1\dots l$ **do** $[y^\iota]_k\leftarrow\zeta^{[L+1]}(\mathcal{X}^{h,\iota}_{t_i}; A_{i,k}, b_{i,k})$
      else
        $y^\iota\leftarrow 0$
      end
      $\varrho^\iota\leftarrow\zeta^{[L+1]}(\mathcal{X}^{h,\iota}_{t_i}; A_{i,\mathbb{ES}}, b_{i,\mathbb{ES}})$
      $\chi^\iota\leftarrow\chi^\iota + f(t_i, \mathcal{X}^{h,\iota}_{t_i}, y^\iota, \varrho^\iota)\,\Delta t_{i+1}$
    end
    foreach $k=1\dots l$ **do**
      $A_{i,k}, b_{i,k}\leftarrow\text{BaseAlg}(\{(\mathcal{X}^{h,\iota}_{t_i}, [\chi^\iota]_k), \iota\in\mathcal{I}\}, \mathcal{B}, E, \eta, A_{i+1,k}, b_{i+1,k}, \mathcal{L}^2)$
    // Update M_{current} to have
    // $M_{\text{current},i} = Y^{p,h}_{t_n} - Y^{p,h}_{t_i} + \sum_{k=i}^{n-1} f(t_k, \mathcal{X}^h_{t_k}, Y^{p,h}_{t_k}, \rho^{p,h}_{t_k})\,\Delta t_{k+1}$
    // where p is the current Picard iteration
    foreach $\iota\in\mathcal{I}$ **do** $M^\iota_{i,\text{current}}\leftarrow M^\iota_{i+1,\text{current}} + \chi^\iota - y^\iota$
  end
  foreach $i=n-1\dots1$ **and** $i\in\mathcal{I}$ **do** $M^\iota_{i,\text{prev}}\leftarrow M^\iota_{i,\text{current}}$
end

4.3.3 A Posteriori Analysis of the Regression Error

A well-established BSDE spatial error analysis strategy consists in analysing the accumulation, over (discrete) time i decreasing from $n-1$ to 0, of three error components [Gobet, 2016]: (i) a *bias* between (the function representing) Y_i^h (as $u_i^h(X_i^h)$ for a suitable measurable function $u_i(x)$) and the hypothesis space of functions in which $\hat{Y}_i^h$ is sought after, (ii) a *"variance"* in the sense of the regression estimation error, and (iii) a term of *propagation* at time i of the error at time $i+1$. This is at least the strategy in the standard case where the embedded conditional expectations are estimated by linear least-squares regressions that can be performed exactly, for instance by singular value decomposition [Gobet, 2016]. Neural net parameterizations for the targeted functions (conditional expectations or expected shortfalls in the case of our ABSDEs) instead lead to "nonlinear regressions" that can only be performed by numerical, nonconvex minimization. When state-of-the-art, fine-tuned, Adam variants of stochastic gradient descents are used in this regard, the ensuing minimization can be very efficient numerically. However, there is no known learning algorithm solving such nonconvex minimization problems with an a priori error bound. Hence, when the learning iteration terminates, we do not have any guarantee on the quality of the approximation. In other words, there is a fourth *numerical minimization* error component on top of the three other ones in the above and this fourth error component cannot be controlled ex ante.

However, we can still assess the local regression error of the schemes by an *a posteriori Monte Carlo procedure*, as follows. For concreteness we assume $l=1$ and a uniform time step Δt, in the XVA motivated case[4.18]

$$\bar{t} = (t+1) \wedge T \text{ and } \Phi_{\bar{t}}(M) = M_{\bar{t}} - M_t \tag{4.19}$$

The following developments are done in the case of the explicit scheme, but similar computations would yield similar outputs in the case of the implicit/Picard scheme. Letting $m = \lfloor \frac{1}{\Delta t} \rfloor$, the time-explicit fully discrete scheme (4.17) here reduces to $\hat{Y}_{t_n}^h = \phi(\mathcal{X}_T^h)$, $\hat{\rho}_{t_n}^h = 0$ and, for k decreasing from $n-1$ to 0,

$$\begin{aligned}
\hat{Y}_{t_k}^h &= \hat{\mathbb{E}}_{t_k}[\hat{Y}_{t_{k+1}}^h + f(t_k, \mathcal{X}_{t_k}^h, \hat{Y}_{t_{k+1}}^h, \hat{\rho}_{t_{k+1}}^h)\,\Delta t] \\
\hat{\rho}_{t_k}^h &= \widehat{\mathbb{E}\mathbb{S}}_{t_k}[\hat{Y}_{t_{(k+m)\wedge n}}^h - \hat{Y}_{t_k}^h + \sum_{i=k}^{(k+m-1)\wedge(n-1)} f(t_i, \mathcal{X}_{t_i}^h, \hat{Y}_{t_{i+1}}^h, \hat{\rho}_{t_{i+1}}^h)\,\Delta t].
\end{aligned}$$

We also define the following auxiliary scheme: $\tilde{Y}_{t_n}^h = \phi(\mathcal{X}_T^h)$, $\tilde{\rho}_{t_n}^h = 0$ and, for k decreasing from $n-1$ to 0,

$$\begin{aligned}
\tilde{Y}_{t_k}^h &= \mathbb{E}_{t_k}[\hat{Y}_{t_{k+1}}^h + f(t_k, \mathcal{X}_{t_k}^h, \hat{Y}_{t_{k+1}}^h, \hat{\rho}_{t_{k+1}}^h)\,\Delta t] \\
\tilde{\rho}_{t_k}^h &= \mathbb{E}\,\mathbb{S}_{t_k}[\hat{Y}_{t_{(k+m)\wedge n}}^h - \hat{Y}_{t_k}^h + \sum_{i=k}^{(k+m-1)\wedge(n-1)} f(t_i, \mathcal{X}_{t_i}^h, \hat{Y}_{t_{i+1}}^h, \hat{\rho}_{t_{i+1}}^h)\,\Delta t].
\end{aligned}$$

Let

$$\epsilon_{t_i} = |\tilde{Y}_{t_i}^h - \hat{Y}_{t_i}^h|. \tag{4.20}$$

Proceeding as in [Abbas-Turki et al., 2021](Section 5.1), one can estimate $\mathbb{E}[\epsilon_{t_k}^2]$ *without computing* $\tilde{Y}_{t_k}^h$, by Monte Carlo using a so-called twin simulation trick, based on the following identity involving two copies $\hat{Y}_{t_{k+1}}^{h,1}$ and $\hat{Y}_{t_{k+1}}^{h,2}$ of $\hat{Y}_{t_{k+1}}^h = u_{k+1}^h(\mathcal{X}_{t_{k+1}}^h)$ and $\hat{\rho}_{t_{k+1}}^{h,1}$ and

4.18. cf. Section 4.4.

$\hat{\rho}^{h,2}_{t_{k+1}}$ of $\hat{\rho}^h_{t_{k+1}} = v^h_{k+1}(\mathcal{X}^h_{t_{k+1}})$, where u^h and v^h are the regressed functional forms of $\hat{Y}^h_{t_{k+1}}$ and $\hat{\rho}^h_{t_{k+1}}$, with all copies independent conditionally[4.19] on $\mathcal{X}^h_{t_k}$:

$$\begin{aligned}\mathbb{E}[\epsilon^2_{t_k}] = \mathbb{E}[|\hat{Y}^h_{t_k}|^2 - 2\,\hat{Y}^h_{t_k}\,(\hat{Y}^h_{t_{k+1}} + f(t_k, \mathcal{X}^h_{t_k}, \hat{Y}^h_{t_{k+1}}, \hat{\rho}^h_{t_k+1})\,\Delta t) + \\ (\hat{Y}^{h,1}_{t_{k+1}} + f(t_k, \mathcal{X}^h_{t_k}, \hat{Y}^{h,1}_{t_{k+1}}, \hat{\rho}^{h,2}_{t_k+1})\,\Delta t)\,(\hat{Y}^{h,2}_{t_{k+1}} + f(t_k, \mathcal{X}^h_{t_k}, \hat{Y}^{h,2}_{t_{k+1}}, \hat{\rho}^{h,2}_{t_k+1})\,\Delta t)].\end{aligned} \tag{4.21}$$

As detailed in [Barrera et al., 2022](Section 4.4), the $e_{t_i} = |\tilde{\rho}^h_{t_i} - \hat{\rho}^h_{t_i}|$ can also be estimated by twin simulation, without having to compute the $\tilde{Y}^h_{t_k}$ and $\tilde{\rho}^h_{t_k}$.

Proceeding in this way, we obtain an a posteriori Monte Carlo procedure to assess, at least locally in time, the spatial error of our ABSDE numerical schemes. In the case where the error is not good enough, one can improve the stochastic gradient descent, in first attempt, and then act on the hypothesis spaces, e.g., in the case of neural networks, try to train with more layers/units or better architectures.

Remark 4.7. In the standard BSDE case where $f_k(t, x, y, \varrho) = f_k(t, x, y)$, i.e. there is no dependence of the coefficient of the BSDE on ρ, by Λ_f-Lipschitz continuity of $f_k(t, x, y)$ with respect to y, we have:

$$\mathbb{E}[|Y^h_{t_k} - \tilde{Y}^h_{t_k}|] \le (1 + \Lambda_f\,\Delta t)\,\mathbb{E}[|Y^h_{t_{k+1}} - \hat{Y}^h_{t_{k+1}}|]$$

and the triangular inequality yields

$$\mathbb{E}[|Y^h_{t_k} - \hat{Y}^h_{t_k}|] \le (1 + \Lambda_f\,\Delta t)\,\mathbb{E}[|Y^h_{t_{k+1}} - \hat{Y}^h_{t_{k+1}}|] + \mathbb{E}[|\hat{Y}^h_{t_k} - \tilde{Y}^h_{t_k}|]. \tag{4.22}$$

Hence, using the inequality $\mathbb{E}[\epsilon_{t_k}] \le \sqrt{\mathbb{E}[\epsilon^2_{t_k}]}$:

$$\mathbb{E}[|Y^h_{t_k} - \hat{Y}^h_{t_k}|] \le \sum_{i=k}^{n-1} (1 + \Lambda_f\,\Delta t)^{i-k} \sqrt{\mathbb{E}[\epsilon^2_{t_i}]}. \tag{4.23}$$

Each $\mathbb{E}[\epsilon^2_{t_i}]$ in (4.23) can be computed by twin Monte Carlo based on (4.21)[4.20], for two copies $\hat{Y}^{h,1}_{t_{k+1}}$ and $\hat{Y}^{h,2}_{t_{k+1}}$ of $\hat{Y}^h_{t_{k+1}}$ that are independent conditionally on $\mathcal{X}^h_{t_k}$.

In the anticipated case, the analogous propagation of the local regression error terms ϵ_{t_i} and e_{t_i} into global regression error controls for $\mathbb{E}[|Y^h_{t_k} - \hat{Y}^h_{t_k}|]$ and $\mathbb{E}[|\rho^h_{t_k} - \hat{\rho}^h_{t_k}|]$ is currently under study.

4.4 XVA Application

We consider a bank dealing financial derivatives with multiple counterparties indexed by c, with default times $\tau^{(c)}$, where all portfolios are uncollateralized with zero recovery in the case of defaults (all assumed instantaneously settled). For notational simplicity we assume no contractual cash flows between the bank and client c at $\tau^{(c)}$. We denote by $T > 0$ the final maturity of the derivative portfolio of the bank and by $\mathrm{MtM}^{(c)}$ the aggregated mark-to-market process (counterparty-risk-free valuation) of the portfolio of the bank with counterparty c. The bank, with risky funding spread process γ, is required to maintain a minimum amount of capital at risk, at the level of an economic capital (EC) defined below as an expected shortfall of the bank trading loss over one year at a confidence level $\alpha \in (\frac{1}{2},$

4.19. hence, one simply simulates two independent realizations of $\mathcal{X}^h_{t_{k+1}}$ given the same starting point $\mathcal{X}^h_{t_k}$ and then takes their images by the learned functionals u^h_{k+1} and v^h_{k+1}.

4.20. ignoring the $\hat{\rho}^h_{t_k+1}$ there.

1). The bank is assumed perfectly hedged in terms of market risk, hence its trading loss reduces to the one of the CVA and FVA desks. For the sake of brevity in notation, we omit the discountings at the risk-free rate in the equations (in other terms, we use the risk-free asset as a numéraire), while preserving them in the numerical codes. We assume a KVA risk premium, i.e. bank shareholders earn a hurdle rate $r>0$ on their capital at risk. Finally we assume that the bank can use its capital as a risk-free funding source [Crépey et al., 2020].

As in [Albanese et al., 2021], we define the CVA (credit valuation adjustment), FVA (funding valuation adjustment), EC (economic capital) and KVA (capital valuation adjustment) via the following continuous-time coupled XVA equations, where $J_t^{(c)}=\mathbb{1}_{\{t<\tau^{(c)}\}}$ (so that $-d\,J_t^{(c)}=\delta_{\tau^{(c)}}(d\,t)$, with $\delta_{\tau^{(c)}}$ the Dirac measure at time $\tau^{(c)}$):

$$\begin{aligned}
\mathrm{CVA}_t &= \sum_c \mathbb{E}_t\left[\int_t^T (\mathrm{MtM}_s^{(c)})^+\,\delta_{\tau^{(c)}}(d\,s)\right]\\
\mathrm{FVA}_t &= \mathbb{E}_t\left[\int_t^T \gamma_s\left(\sum_c J_s^{(c)}\,\mathrm{MtM}_s^{(c)} - \mathrm{CVA}_s - \mathrm{FVA}_s - \max\left(\mathrm{EC}_s, \mathrm{KVA}_s\right)\right)^+ d\,s\right]\\
\mathrm{KVA}_t &= \mathbb{E}_t\left[\int_t^T r\,e^{-r(s-t)}\max\left(\mathrm{EC}_s, \mathrm{KVA}_s\right) d\,s\right]
\end{aligned} \tag{4.24}$$

where, with $\bar{t}=(t+1)\wedge T$,

$$\mathrm{EC}_t = \mathbb{ES}_t\left[L_{\bar{t}} - L_t\right], \tag{4.25}$$

in which the loss process L satisfies (starting from 0 at time 0)

$$\begin{aligned}
\mathrm{d}L_t &= \mathrm{dCVA}_t + \sum_c (\mathrm{MtM}_t^{(c)})^+\,\delta_{\tau^{(c)}}(d\,t) + \mathrm{dFVA}_t\\
&\quad + \gamma_s\left(\sum_c J_t^{(c)}\,\mathrm{MtM}_t^{(c)} - \mathrm{CVA}_t - \mathrm{FVA}_t - \max\left(\mathrm{EC}_t, \mathrm{KVA}_t\right)\right)^+ d\,t.
\end{aligned} \tag{4.26}$$

All random variables $J_t^{(c)}, \mathrm{MtM}_t^{(c)}$, CVA_t defined by the first line in (4.24), as well the (pre-)default intensity γ_t of the bank, are assumed to be $\sigma(\mathcal{X}_t)$-measurable random variables. The FVA and the KVA equations in (4.24) can then be written in the form (4.7), for

$$\begin{aligned}
Y_t &= \begin{bmatrix} \mathrm{FVA}_t \\ e^{-rt}\,\mathrm{KVA}_t \end{bmatrix}\\
f_k(t,x,y,\varrho) &= \begin{bmatrix} \gamma_k(t,x)\left(\sum_c J_k^{(c)}(t,x)\mathrm{MtM}_k^{(c)}(t,x) - \mathrm{CVA}_k(t,x) - y^1 - \max\left(\varrho, e^{rt}\,y^2\right)\right) \\ r\max\left(e^{-rt}\,\varrho, y^2\right) \end{bmatrix}\\
\Phi_{\cdot}(M) &= \int_{\cdot}^{\bar{\cdot}}\left(\mathrm{dCVA}_t + \sum_c (\mathrm{MtM}_t^{(c)})^+\,\delta_{\tau^{(c)}}(d\,t) + d\,M_t^1\right).
\end{aligned} \tag{4.27}$$

where we denote $Z_k(t,x)=\mathbb{E}[Z_t|X_t=x, J_t=k]$, for any process Z. In fact, for (Y,M) solving the ABSDE (4.7) corresponding to the specification (4.26), we have

$$d\,M_t^1 = \mathrm{dFVA}_t + \gamma_t\left(\sum_c J_t^{(c)}\,\mathrm{MtM}_t^{(c)} - \mathrm{CVA}_t - \mathrm{FVA}_t - \max\left(\mathrm{EC}_t, \mathrm{KVA}_t\right)\right)^+ d\,t,$$

hence, by (4.26),

$$\Phi_{\bar{t}}(M) = L_{\bar{t}} - L_t.$$

Note that

$$|\Phi(t;\mathbf{x},\mathbf{m})-\Phi(t;\mathbf{x},\mathbf{m}')|\leq|\mathbf{m}_{\bar{t}}-\mathbf{m}'_{\bar{t}}|,$$

so that Assumption 4.2(iii) holds with $\Lambda_\Phi=1$. The other conditions in Assumption 4.2 are not hard to check.

In view of the first line in (4.24), the CVA process can be estimated by nonparametric regression in space, at each grid pricing time t_i, of Monte Carlo simulated values of the forward process $\mathcal{X}$[4.21] at t_i. The exercise is made delicate by the hybrid nature of $\mathcal{X}$, which includes both diffusive (market risk) and discrete (default risk) components. This difficulty can be solved by adopting the hierarchical simulation scheme of Abbas-Turki et al., 2021, whereby an optimized number of default trajectories is simulated conditional on each simulated trajectory of the market. As a consequence, hereafter, the CVA process is treated as a given (already estimated) process.

The FVA, EC, and KVA are challenging due to their coupling via the loss process L. However, this coupling can be overcome by a combination of time discretization schemes and (or not) Picard iterations as presented in a more general context in Section 4.3. The specification of both schemes to the XVA case, as well as a direct variant of the implicit scheme[4.22] which is also available in this case, are detailed in Section 4.A. The embedded regressions and quantile regressions are implemented using a neural network of one hidden layer with 38 neurons[4.23], the way detailed in Section 4.3.2.

Remark 4.8. In practice, for variance reduction purposes, we use a default intensities based reformulation of the CVA, instead of the definition based on default indicators in (4.24). The default indicators based CVA was used in [Abbas-Turki et al., 2021] mainly for benchmarking reasons. However, we still use the hierarchical simulation scheme of [Abbas-Turki et al., 2021] to simulate several default paths given each realization of the diffusion processes, in order to help with learnings where we do not have the convenience of using default intensities (given in particular the presence of default terms in the loss (4.26) that occur nonlinearly in EC computations).

4.4.1 Numerical Results

For our numerical experiments, we assume 10 economies, each represented by a short-rate process with Vasicek dynamics, 9 cross-currency rate processes with log-normal dynamics (and stochastic interest rates), 8 counterparties each with a stochastic default intensity process following CIR dynamics, which in total yields 28 stochastic risk factors when accounting for the spread of the bank which is also assumed to be driven by a CIR process.

In order to have analytic mark-to-markets, we assume that the portfolio of the bank is comprised of 100 interest rate swaps, all of whom are assumed to be at-par at time 0. The characteristics of the swaps (notional, maturity, counterparty and currency) are drawn randomly, and in particular their maturities are between 0.9375 and 10 years, hence $T=10$.

For the purpose of a time discretization analysis, we consider multiple time discretizations $(h^{(\iota)})_\iota$ such that:

$$h^{(\iota)}=\{t_i^{(\iota)}:=i\frac{T}{2^\iota},i=0,\dots,2^\iota\}$$

(nested with respect to ι, in order to have common cash-flow dates over the different time discretizations). In figures 4.1, 4.2 and 4.5, we tested for $\iota\in\{5,6,7,8\}$ (τ in the figures then corresponds to $\frac{T}{2^8}$).

4.21. or its time-discretized version $\mathcal{X}^h$.

4.22. without need for Picard iterations, but at the cost of a shift of time step in EC.

4.23. for experimentation with the network architecture see [Albanese et al., 2021](Section 5.1, Figure 2).

Figures 4.1-4.2 and Table 4.1 show the convergence of the Picard iterates, using Algorithm 4.3, of the FVA process toward a solution visually very close, already for $j=4$, to the one provided by the explicit scheme in Figure 4.5, but at a higher computational cost. We also attempted to improve the Picard scheme using less epochs and reusing weights across Picard iterations as discussed in Remark 4.6, but the resulting iterates exhibited a slight instability when using finer time step, i.e. for $\iota=8$: see the bottom graphs in Figures 4.3-4.4 and Table 4.2. A possible explanation is that Algorithms 4.5 and 4.3 accumulate a large number of stochastic gradient descents via the weights reused across pricing times, with integrands involving the same stochastic processes just observed at different but close time steps. With weights reused across Picard iterations in the modified Picard scheme, instead, the integrands are not guaranteed to be close. The results obtained in Figures 4.1 and 4.2 and Table 4.1 using the standard Picard scheme with reuse of the weights across time steps, i.e. Algorithm 4.3, support this conjecture. These results mutually validate the explicit and the standard Picard scheme against each other. However, the standard Picard scheme is not competitive with respect to the explicit scheme in terms of computation times (cf. Remark 4.6).

In conclusion, the explicit scheme dominates Picard iterations, whether that a common computation time is allocated to both schemes and the explicit scheme is more accurate, or that both schemes converge but this is then at the cost of $j=3$-4 times longer computations in the case of the Picard iterations (for reference the explicit scheme takes 7mins48secs using the finest time grid, *i.e.* $\iota=8$, on an NVidia A100 GPU).

Figures 4.6, 4.7, 4.8 and 4.9 show plots of profiles for respectively the CVA, FVA, KVA and EC when using the explicit scheme. Figure 4.10 illustrates the convergence in time of the scheme. The solid purple curves in Figures 4.11 and 4.12 exhibit the local regression errors $\sqrt{\mathbb{E}[\epsilon_{t_i}^2]}$ of the scheme[4.24] for the FVA and the KVA. The dashed grey curves represent the corresponding L^2 training losses. The comparison between the grey and purple curves shows the benefit of our a posteriori Monte Carlo local regression error estimate with respect to the L^2 training losses that would be used as a naive (but overconservative) error estimate.

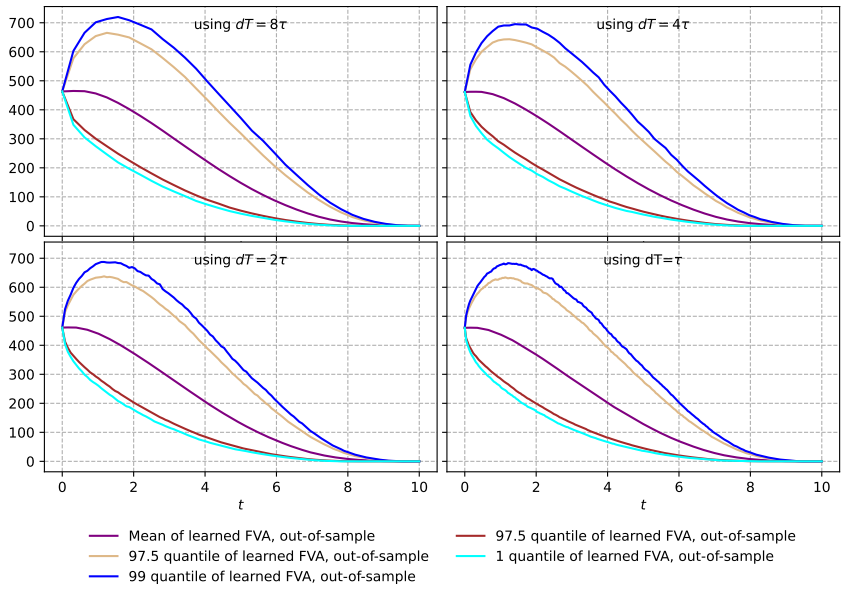

Figure 4.1. FVA profiles obtained after $j=1$ Picard iteration for the implicit scheme.

4.24. cf. (4.20).

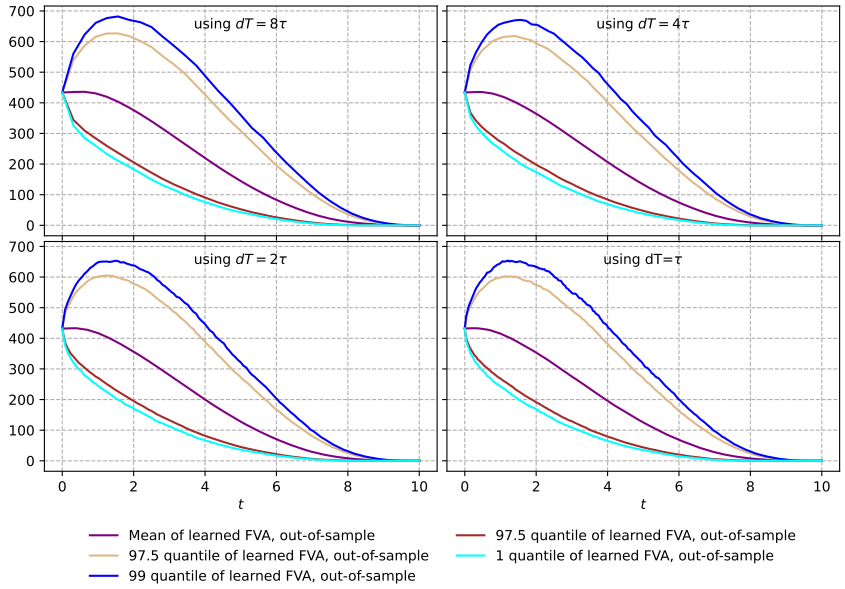

Figure 4.2. FVA profiles obtained after $j = 4$ Picard iteration for the implicit scheme.

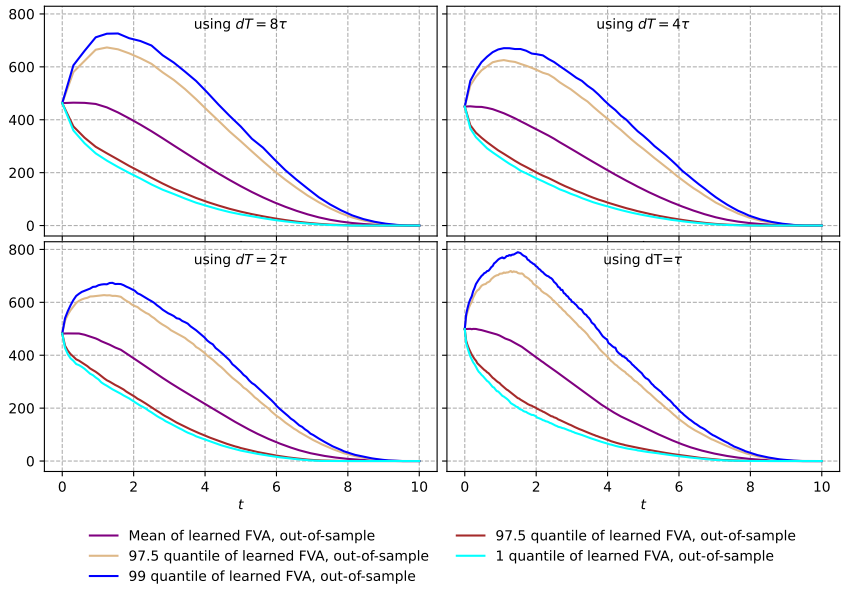

Figure 4.3. FVA profiles obtained after $j = 1$ Picard iteration for the implicit scheme, using less SGD steps and reusing the weights of the previous Picard iteration at each learning.

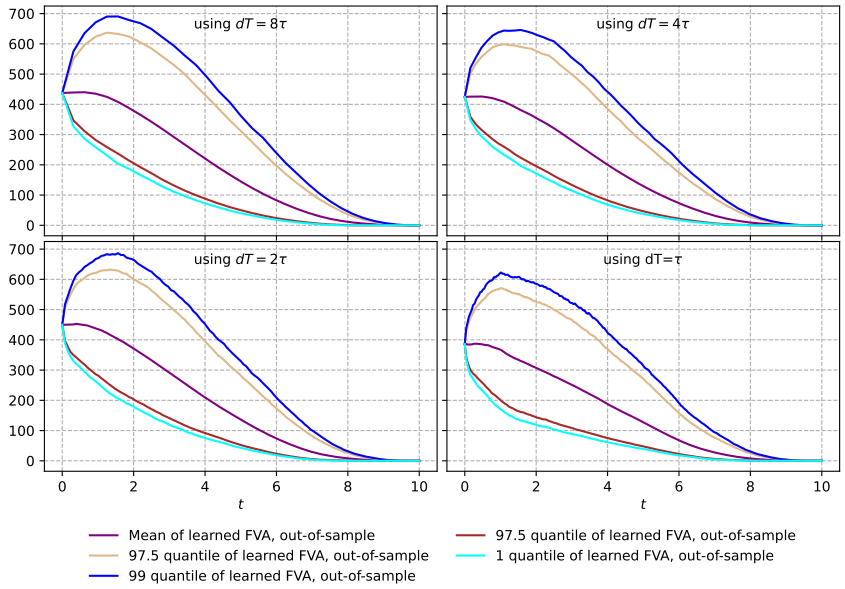

Figure 4.4. FVA profiles obtained after $j = 4$ Picard iterations for the implicit scheme, using less SGD steps and reusing the weights of the previous Picard iteration at each learning.

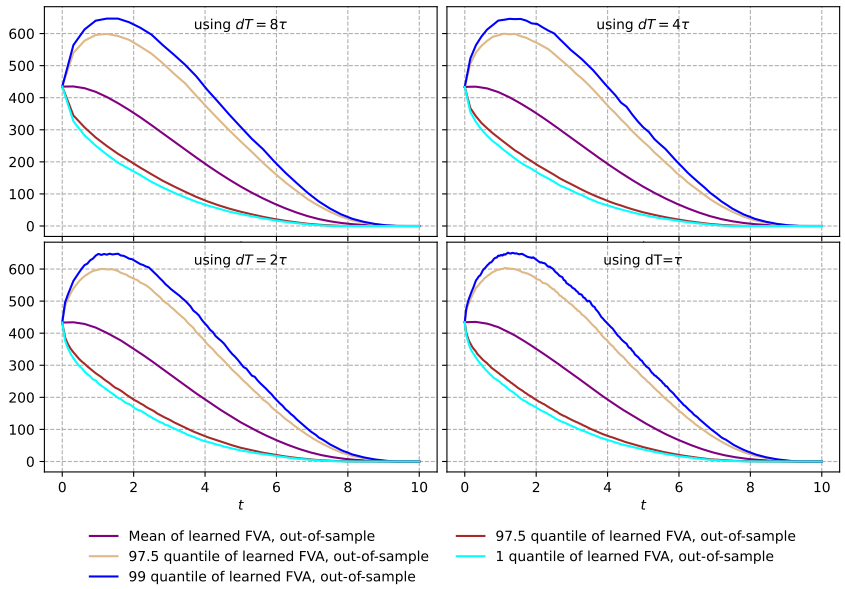

Figure 4.5. FVA profiles using an explicit scheme.

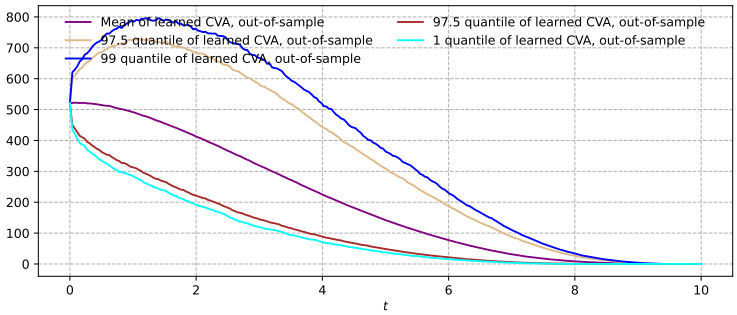

Figure 4.6. CVA profiles using an explicit scheme and a fine time discretization ($\iota = 10$).

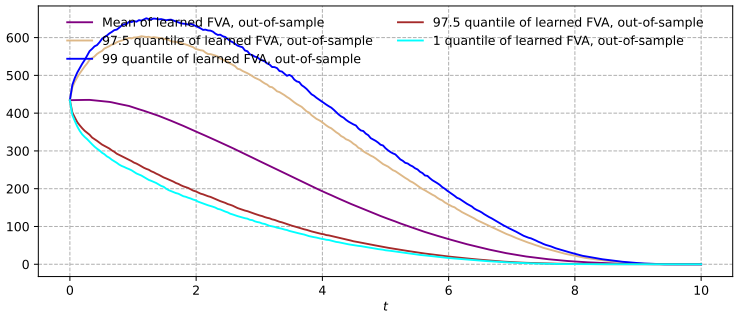

Figure 4.7. FVA profiles using an explicit scheme and a fine time discretization ($\iota = 10$).

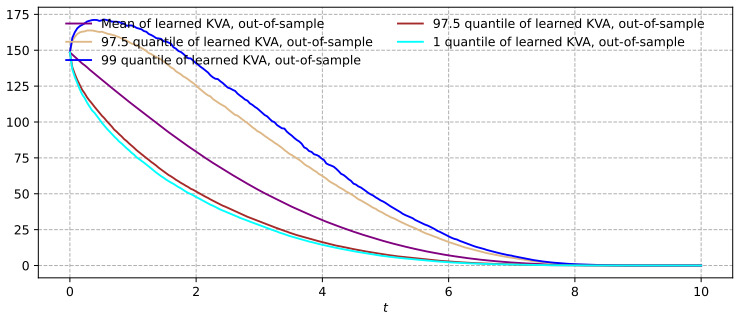

Figure 4.8. KVA profiles using an explicit scheme and a fine time discretization ($\iota = 10$).

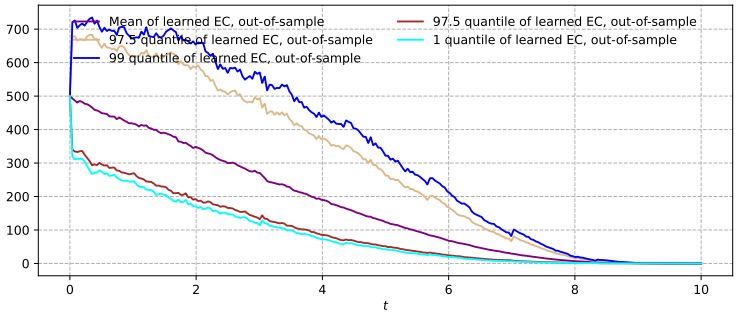

Figure 4.9. EC profiles using an explicit scheme and a fine time discretization ($\iota = 10$).

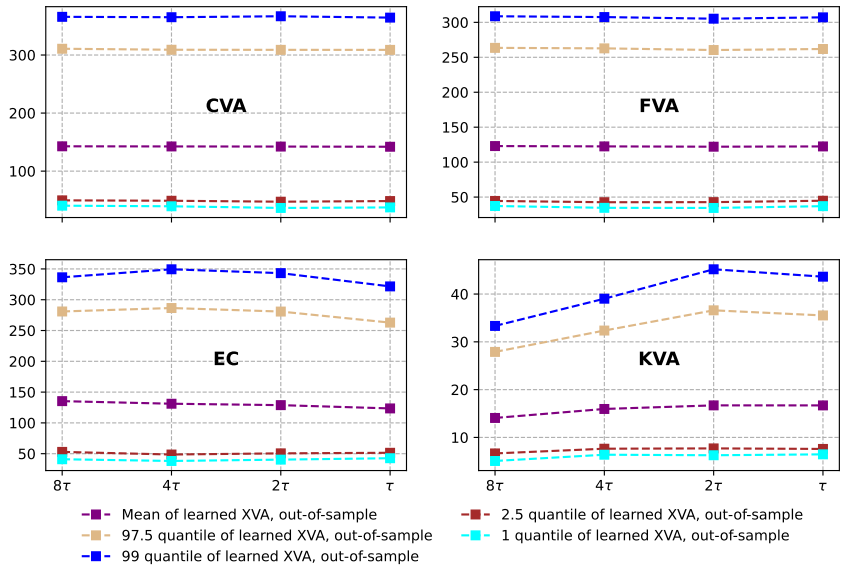

Figure 4.10. Mean and quantiles of learned CVA, FVA, KVA and EC at $t = \frac{T}{2}$ for different sizes of the time step.

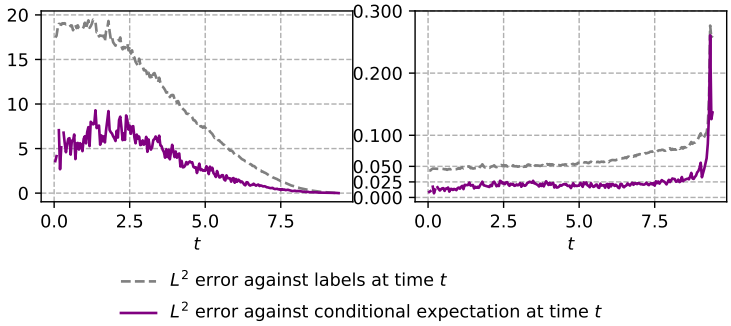

Figure 4.11. Local regression errors $\sqrt{\mathbb{E}[(\epsilon^{\text{fva}}_{t_i})^2]}$ (solid purple) vs. L^2 training losses (dashed grey). **Left panel:** raw errors. **Right panel:** errors normalized by the $\widehat{\text{FVA}}^h_{t_i}$.

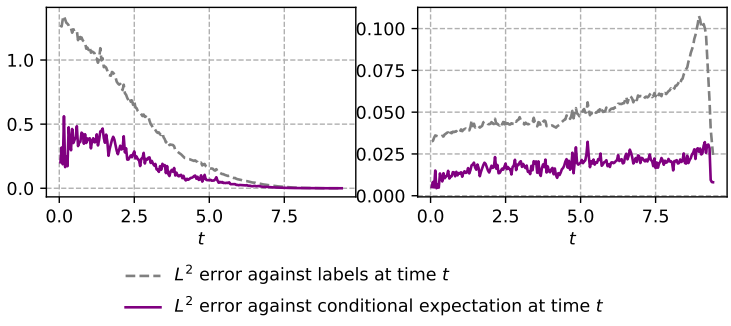

Figure 4.12. Local regression errors $\sqrt{\mathbb{E}[(\epsilon^{\text{kva}}_{t_i})^2]}$ (solid purple) vs. L^2 training losses (dashed grey). Left panel: raw errors. Right panel: errors normalized by the $\widehat{\text{KVA}}^h_{t_i}$.

	$j=1$	$j=2$	$j=3$	$j=4$	Explicit
$h=\frac{T}{2^5}$	463.279938	433.832031	434.391296	433.753998	434.65167
$h=\frac{T}{2^6}$	461.329926	433.141876	434.036011	433.835052	433.60974
$h=\frac{T}{2^7}$	461.031097	432.506531	433.631531	431.789215	433.18683
$h=\frac{T}{2^8}$	460.326050	433.123596	431.992859	432.098328	434.29538

Table 4.1. FVA_0 under the Picard iteration scheme with reuse of weigthts across time steps (*i.e.* Algorithm 4.3) vs. the explicit scheme.

	$j=1$	$j=2$	$j=3$	$j=4$	Explicit
$h=\frac{T}{2^8}$	498.9785	416.31674	464.44170	386.07004	434.295380
$h=\frac{T}{2^7}$	481.9461	443.68940	440.41504	449.64874	433.186829
$h=\frac{T}{2^6}$	449.9881	435.35910	429.49142	424.67087	433.609741
$h=\frac{T}{2^5}$	462.9291	438.05840	435.67453	437.58957	434.651672

Table 4.2. FVA_0 under the modified Picard vs the explicit scheme.

4.5 Conclusion

The recent and fastly growing machine learning literature on the solution of high-dimensional nonlinear BSDEs(/PDEs) contains, at least, two branches. The first one, in the line of E et al., 2017, consists in learning together the time-0 value of the solution and the gradient-process of the latter through a single learning task. Examples in the XVA space include Henry-Labordère, 2017 or Gnoatto et al., 2021. The former reference provides insightful PDE views on the CVA and MVA, while it is very tempting to adopt an approach, as in the second reference, where the XVAs and their sensitivities are obtained simultaneously. However, the equations considered by Henry-Labordère, 2017 are only very distantly related to actual XVA equations. The XVA equations of Gnoatto et al., 2021 are more realistic, but they are still restricted to computations at the level of one netting set (or client) of the bank, and handled by reduction of filtration in the line of Crépey and Song, 2015, so that the default times ultimately disappear from the equations. Such an approach does not leverage to several clients and default times of the bank, that enter the XVA equations in a nonlinear fashion (and therefore have to be simulated). The second stream of literature, see e.g. Huré et al., 2020, learns the solution time step after time step (in backward time), much like in classical dynamic programming algorithms, except that modern machine learning techniques are used for solving the local equations which then arise at each successive decreasing pricing time step.

In the case of our ABSDE XVA equations, on the one hand, the global approach would not be viable on realistic problems stated at the portfolio level, because of the huge RAM memory demand of the corresponding global training task. On the other hand, the local approach benefits from a particular synergy between the successive local training tasks involved. In fact, as all XVA equations are endowed with zero terminal conditions, the

variance of the labels, i.e. the cash flows entering the successive learning tasks as input data at the decreasing pricing time steps, increases progressively (pricing time step after pricing time step), whereas the variance of the features, i.e. the risk factors, decreases. As a result, the difficulty of the training tasks gradually increases throughout the course of the algorithm[4.25]. But the next training task also greatly (and increasingly) benefits from all previous ones, via the use of the weights trained at a time step as initialization for the weights at the next time step. This is probably one of the reasons behind the robustness of the local machine learning approach on our problem—provided the hierarchical simulation technique of Abbas-Turki et al., 2021 and the best practice risk measure estimators of Barrera et al., 2022 are used,—when a global approach would fail on unsolvable memory occupation issues.

Regarding the comparison between the direct explicit scheme and the implicit scheme solved by Picard iteration, the explicit scheme emerges from the present study as the preferred alternative (at the level of this paper, implementing the two schemes was of course useful from a mutual numerical validation viewpoint[4.26]).

From an algorithmic viewpoint, work in progress aims at demonstrating how the regression-based XVA simulation framework of the present paper can be leveraged to also encompass XVA sensitivities, or hedging ratios more generally. Note that AAD sensitivities computational techniques à la Baydin et al., 2018; Savine, 2018 are not a viable alternative in our setup, where the XVA metrics are the output of optimization training procedures (AAD sensitivities techniques can only be available in much more rudimentory XVA setups). From a mathematical viewpoint, the establishment of a Feynman-Kac representation for the limiting ABSDE (4.7), as well as the study of the time-consistency of both schemes, and of the propagation of the local into global spatial regression errors[4.27], are challenging open issues.

4.A XVA Numerical Schemes

We assume a uniform time step $\Delta t = h$ to alleviate the notation. By least squares (resp. quantile) regressions below, we actually mean *neural net* least squares (resp. quantile)

4.25. In order to see this in a simplified setup, consider linear regression instead of neural networks. When close to $t = 0$, the variances of the features tend to 0, which leads to ill-conditioned covariance matrices, while the variance of the labels increases, which makes the regression even more unstable.

4.26. Moreover, for a suitable initialization, the output of the first Picard iteration (2 or 3 are typically enough in practice) is interesting in itself from a financial interpretation viewpoint in an XVA setup, as this first iteration corresponds to the XVA numbers ignoring the possibility to use capital at risk for variation margin funding purposes [Albanese et al., 2017].

4.27. cf. the end of Remark 4.7.

regressions, in the sense detailed in Section 4.3.2.

4.A.1 Explicit scheme

Here we use the following time-discretization, skipping indices $\cdot^h$ to alleviate the notation and writing $\bar{t}_k = t_{k+1/h} \wedge t_n$: $\mathrm{CVA}_{t_n} = \mathrm{FVA}_{t_n} = \mathrm{KVA}_{t_n} = 0$ followed by, for $k = n-1 \cdots 0$,

$$
\begin{aligned}
\mathrm{CVA}_{t_k} &= \mathbb{E}_{t_k}\left[\sum_c \sum_{k \le i \le n-1} (\mathrm{MtM}^{(c)}_{t_{i+1}})^+ \mathbb{1}_{\{t_i < \tau^{(c)} \le t_{i+1}\}}\right] \\
\mathrm{FVA}_{t_k} &= \mathbb{E}_{t_k}\Big[\mathrm{FVA}_{t_{k+1}} + \\
&\qquad h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}^{(c)}_{t_k} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_{k+1}} - \max\left(\mathrm{EC}_{t_{k+1}}, \mathrm{KVA}_{t_{k+1}}\right)\right)^+\Big] \\
\mathrm{KVA}_{t_k} &= \exp\left(-r\,h\right) \mathbb{E}_{t_k}\left[\mathrm{KVA}_{t_{k+1}} + r\,h \max\left(\mathrm{EC}_{t_{k+1}}, \mathrm{KVA}_{t_{k+1}}\right)\right] \\
\mathrm{EC}_{t_k} &= \mathbb{ES}_{t_k}\left[L_{\bar{t}_k} - L_{t_k}\right], \text{ where} \\
L_{t_{k+1}} - L_{t_k} &= \mathrm{CVA}_{t_{k+1}} - \mathrm{CVA}_{t_k} + (\mathrm{MtM}^{(c)}_{t_k})^+ \mathbb{1}_{\{t_k < \tau^{(c)} \le t_{k+1}\}} + \mathrm{FVA}_{t_{k+1}} - \mathrm{FVA}_{t_k} + \\
&\qquad h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}^{(c)}_{t_k} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_{k+1}} - \max\left(\mathrm{EC}_{t_{k+1}}, \mathrm{KVA}_{t_{k+1}}\right)\right)^+.
\end{aligned}
$$

The explicit scheme naturally lifts the coupling visible in (4.24)–(4.26) between EC, KVA, and FVA. Let $k \in \{1, n-1\}$ and assume one has already estimated the introduced XVAs at all times $\{k+1, \ldots, n\}$. We compute:

1. CVA, by a least-squares regression of $\sum_c \sum_{k \le i \le n-1} (\mathrm{MtM}^{(c)}_{t_i})^+ \mathbb{1}_{\{t_i < \tau^{(c)} \le t_{i+1}\}}$ (or equivalent variance-reduced cash flows formulated in terms of the default *intensities* as explained after (4.26)) against the market risk factors at time t_k;

2. FVA_{t_k}, through a least-squares regression of

$$
\begin{aligned}
&\mathrm{FVA}_{t_{k+1}} + \\
&h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}^{(c)}_{t_k} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_{k+1}} - \max\left(\mathrm{EC}_{t_{k+1}}, \mathrm{KVA}_{t_{k+1}}\right)\right)^+
\end{aligned}
$$

   against all risk factors (market risk factors and client default indicators) at time t_k;

3. KVA_{t_k}, through a least-squares regression of

$$
\exp\left(-r\,h\right)\left(\mathrm{KVA}_{t_{k+1}} + r\,h \max\left(\mathrm{EC}_{t_{k+1}}, \mathrm{KVA}_{t_{k+1}}\right)\right)
$$

   against all risk factors at time t_k;

4. EC_{t_k}, through quantile regression of $L_{\bar{t}_k} - L_{t_k}$ followed by a least-squares regression to deduce the expected shortfall as detailed in Section 4.3.2, both regressions being against all the risk factors at time t_k.

4.A.2 Picard scheme

We define and compute the CVA as in the explicit scheme. For the rest of the XVAs, we introduce Picard iterations, starting from $\mathrm{FVA}^{(0)} = \mathrm{KVA}^{(0)} = 0$ followed by, for increasing

$j \geq 1$: $\mathrm{FVA}_{t_n}^{(j)} = \mathrm{KVA}_{t_n}^{(j)} = 0$ and, for $k = n-1 \cdots 0$,

$$\begin{aligned}
\mathrm{FVA}_{t_k}^{(j)} &= \mathbb{E}_{t_k}\Big[\mathrm{FVA}_{t_{k+1}}^{(j)} + \\
&\qquad h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}_{t_k}^{(c)} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_k}^{(j-1)} - \max\left(\mathrm{EC}_{t_k}^{(j-1)}, \mathrm{KVA}_{t_k}^{(j-1)}\right)\right)^+\Big] \\
\mathrm{KVA}_{t_k}^{(j)} &= \exp\left(-r\,h\right) \mathbb{E}_{t_k}\left[\mathrm{KVA}_{t_{k+1}}^{(j)} + r\,h \max\left(\mathrm{EC}_{t_k}^{(j)}, \mathrm{KVA}_{t_k}^{(j-1)}\right)\right] \\
\mathrm{EC}_{t_k}^{(j)} &= \mathbb{ES}_{t_k}\left[L_{\bar{t}_k}^{(j)} - L_{t_k}^{(j)}\right], \text{ where} \\
&\qquad L_{t_{k+1}}^{(j)} - L_{t_k}^{(j)} = \mathrm{CVA}_{t_{k+1}} - \mathrm{CVA}_{t_k} + \left(\mathrm{MtM}_{t_k}^{(c)}\right)^+ \mathbb{1}_{\{t_k < \tau^{(c)} \leq t_{k+1}\}} + \mathrm{FVA}_{t_{k+1}}^{(j)} - \mathrm{FVA}_{t_k}^{(j)} + \\
&\qquad\quad h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}_{t_k}^{(c)} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_k}^{(j-1)} - \max\left(\mathrm{EC}_{t_k}^{(j-1)}, \mathrm{KVA}_{t_k}^{(j-1)}\right)\right)^+.
\end{aligned}$$

In this scheme the coupling between EC, KVA, and FVA is removed by the Picard iterations in j. In the above, assuming that all the $XVA^{(j)}$ have already been computed at times $\{k+1, \ldots, n\}$, we compute:

1. $\mathrm{FVA}_{t_k}^{(j)}$, by least-squares regression of

$$\begin{aligned}
&\mathrm{FVA}_{t_{k+1}}^{(j)} + \\
&\quad h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}_{t_k}^{(c)} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_k}^{(j-1)} - \max\left(\mathrm{EC}_{t_k}^{(j-1)}, \mathrm{KVA}_{t_k}^{(j-1)}\right)\right)^+
\end{aligned}$$

   against the risk factors at time t_k;

2. $\mathrm{KVA}_{t_k}^{(j)}$, through a least-squares regression of

$$\exp\left(-r\,h\right)\left(\mathrm{KVA}_{t_{k+1}}^{(j-1)} + r\,h \max\left(\mathrm{EC}_{t_k}^{(j-1)}, \mathrm{KVA}_{t_k}^{(j-1)}\right)\right)$$

   against all risk factors at time t_k;

3. $\mathrm{EC}_{t_k}^{(j)}$, through quantile regression of $L_{\bar{t}_k}^{(j)} - L_{t_k}^{(j)}$ followed by a least-squares regression to deduce the expected shortfall as detailed in Section 4.3.2, both regressions being against all the risk factors at time t_k.

Remark 4.9. Shifting by one time step the discretization of L in EC, one can also introduce a *hybrid scheme* which is implicit in the FVA and the KVA, while not requiring Picard iterations: $\mathrm{FVA}_{t_n} = \mathrm{KVA}_{t_n} = 0$ followed by, for $k = n-1 \cdots 0$,

$$\begin{aligned}
\mathrm{EC}_{t_k} &= \mathbb{ES}_{t_k}\left[L_{\bar{t}_{k+1}} - L_{t_{k+1}}\right] \\
\mathrm{KVA}_{t_k} &= \exp\left(-r\,h\right)\left(\mathbb{E}_{t_k}[\mathrm{KVA}_{t_{k+1}}] + r\,h \max\left(\mathrm{EC}_{t_k}, \mathrm{KVA}_{t_k}\right)\right) \\
\mathrm{FVA}_{t_k} &= \mathbb{E}_{t_k}\Big[\mathrm{FVA}_{t_{k+1}} + \\
&\qquad h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}_{t_k}^{(c)} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_k} - \max\left(\mathrm{EC}_{t_k}, \mathrm{KVA}_{t_k}\right)\right)^+\Big] \\
L_{t_{k+1}} - L_{t_k} &= \mathrm{CVA}_{t_{k+1}} - \mathrm{CVA}_{t_k} + \left(\mathrm{MtM}_{t_k}^{(c)}\right)^+ \mathbb{1}_{\{t_k < \tau^{(c)} \leq t_{k+1}\}} + \mathrm{FVA}_{t_{k+1}} - \mathrm{FVA}_{t_k} \\
&\quad + h\,\gamma_{t_k}\left(\sum_c \mathrm{MtM}_{t_k}^{(c)} \mathbb{1}_{\{\tau^{(c)} > t_k\}} - \mathrm{CVA}_{t_k} - \mathrm{FVA}_{t_k} - \max\left(\mathrm{EC}_{t_k}, \mathrm{KVA}_{t_k}\right)\right)^+.
\end{aligned}$$

The shift by one time step in the discretization of EC is what makes it possible to define an implicit FVA in this scheme without resorting to Picard iteration. Indeed, FVA_{t_k} in this scheme is the solution of a semi-linear equation in FVA_{t_k} and is given by:

$$\begin{aligned}\mathrm{FVA}_{t_k} \;=\; & \mathbb{E}_{t_k}[\mathrm{FVA}_{t_{k+1}}] + \frac{h\,\gamma_{t_k}}{1+h\,\gamma_{t_k}} \\ & \left(\sum_c \mathrm{MtM}^{(c)}_{t_k}\mathbb{1}_{\{\tau^{(c)}>t_k\}} - \mathrm{CVA}_{t_k} - \mathbb{E}_{t_k}[\mathrm{FVA}_{t_{k+1}}] - \max\left(\mathrm{EC}_{t_k}, \mathrm{KVA}_{t_k}\right)\right)^+.\end{aligned}$$

Similarly, we have for the KVA:

$$\mathrm{KVA}_{t_k} = \exp\left(-r\,h\right)\left(\mathbb{E}_{t_k}[\mathrm{KVA}_{t_{k+1}}] + r\,h\max\left\{\mathrm{EC}_{t_k}, \frac{\exp\left(-r\,h\right)}{1-r\,h\exp\left(-r\,h\right)}\mathrm{KVA}_{t_k}\right\}\right)$$

Chapter 5

Fast Calibration using Complex-Step Sobolev Training

This chapter was single-authored.

We present a new fast calibration technique where we propose to train neural networks to directly perform the orthogonal projection of simulated payoffs of the calibration instrument with randomized model parameters and we enrich the learning task by including pathwise sensitivities of the payoffs with respect to model and product parameters. We show that this particular instance of Sobolev training can be reformulated in a way that requires computing only (stochastic) directional derivatives and we provide a fast, memory-efficient and numerically stable approach to compute those using complex-step differentiation. Our experiment with a fixed-grid piece-wise linear local volatility example demonstrates that one can get competitive price approximations without having to train the neural network on Monte Carlo prices and that both data-set construction and training can be done in reasonable time while preserving a very general framework that can be applied to a broad range of pricing models.

We provide a highly optimized `C++` code based on `libtorch` which includes all the necessary extensions for AAD on holomorphic neural networks on: `https://github.com/BouazzaSE/TorchCSD`.

5.1 Introduction

With the emergence of pricing models such as rough volatility models [Bayer et al., 2016] which do not have closed-form solutions for vanilla option prices and are slow to simulate using Monte Carlo methods, the need to accelerate the calibration of these models arose as one would otherwise have to repeatedly call a slow Monte Carlo pricing procedure during the calibration phase [McCrickerd and Pakkanen, 2018], which is often implemented using iterative optimization algorithms, rendering the models challenging to implement in practice. Notable recent contributions in this area involve the use of Machine Learning methods to provide fast approximations for the pricing function, and then using the *learned* approximation instead of a Monte Carlo pricer during the calibration phase, effectively accelerating the model calibration as the learned approximation is usually fast to compute. In [De Spiegeleer et al., 2018] for instance, the authors propose to approximate the mapping from model and product parameters to vanilla prices using Gaussian Process regression [Rasmussen and Williams, 2006]. Neural network based price approximations have also been proposed [Bayer and Stemper, 2018; Horvath et al., 2021], mainly motivated by the *Universal Approximation Theorem* [Cybenko, 1989; Kidger and Lyons, 2020]. Another approach [Hernandez, 2016] consists in directly approximating the *inverse* function which maps prices to model parameters, effectively skipping the calibration phase altogether.

All of these approaches suffer from the need to construct data-sets of sufficiently accurate Monte Carlo prices which can take days depending on the complexity of the pricing model. While this can be done *off-line* for general and fixed classes of pricing models and thus the time spent can be considered as a one-off upfront cost, it cannot be neglected when having to frequently deal with very custom pricing models where we may have to frequently reconstruct the pricing approximation from scratch. A slow data-set construction process also severely limits the ability to iterate effectively in research and development as one cannot afford the luxury to test out different time discretization schemes, time step sizes, variance reduction techniques or random number generators.

Different from these approaches, we propose a new fast calibration method which does not require generating data-sets of Monte Carlo prices and instead needs only realizations of payoffs corresponding to random model parameters and their path-wise sensitivities. We dub the proposed training procedure *Complex-Step Sobolev training*, which we recognize could be applied to more general problems outside of pricing model calibration and quantitative finance. We highlight the main contributions of this paper as follows:

- We propose to learn a fast vanilla pricing function using a special instance of Sobolev training by learning to orthogonally project both payoffs and path-wise sensitivities of payoffs corresponding to randomized model parameters;
- We propose a method to accelerate our Sobolev training procedure using only directional derivatives in random directions. We also show and prove how to optimally choose the distribution of this random direction such that the induced variance is minimized;
- We accelerate further the computation of the directional derivatives in an AAD-differentiable way using complex-step differentiation while attaining machine precision and preserving numerical stability;
- We give a posteriori L^2 error estimates that can be computed without having access to ground-truth prices;
- Benchmarks and a fixed-grid local volatility example demonstrating the strength of our method are provided. All the simulation codes and the necessary extensions to implement complex-step differentiation in an AAD-differentiable manner with `libtorch` have been made public on https://github.com/BouazzaSE/TorchCSD under a GPLv3 license.

5.2 Learning to Project Payoffs

Consider a stochastic risk-neutral pricing basis $(\Omega, \mathcal{A}, (\mathcal{F}_t)_{t\geq 0}, \mathbb{Q})$ and an $\mathbb{R}^d$-valued standard Brownian motion $(B_t)_{t\geq 0}$ for some $d \geq 1$. Let $f\colon \mathbb{R}_+ \times \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^d$ and $g\colon \mathbb{R}_+ \times \mathbb{R}^d \times \mathbb{R}^n \to \mathbb{R}^{d\times m}$ be two piece-wise continuously differentiable functions such that we have for all $t \geq 0$, $x, y \in \mathbb{R}^d$ and $\xi, \xi' \in \mathbb{R}^n$:

$$\begin{aligned} \|f(t,x,\xi) - f(t,y,\xi)\| + \|g(t,x,\xi) - g(t,y,\xi)\| &\leq K\,\|x-y\| \\ \|f(t,x,\xi) - f(t,x,\xi')\| + \|g(t,x,\xi) - g(t,x,\xi')\| &\leq K\,(1+\|x\|)\,\|\xi-\xi'\| \\ \|f(t,x,\xi)\| + \|g(t,x,\xi)\| &\leq K\,(1+\|x\|) \end{aligned}$$

for some $K>0$, where the norms are the usual Euclidean and Frobenius norms for vectors and matrices respectively. For every $\xi\in\mathbb{R}^n$, we define $(X_t^\xi)_{t\geq 0}$ to be a strong solution to the following multi-dimensional stochastic differential equation:

$$\mathrm{d}X_t^\xi = f(t,X_t,\xi)\,\mathrm{d}t + g(t,X_t,\xi)\,\mathrm{d}B_t \tag{5.1}$$

We assume the same deterministic initial value X_0 for all $\xi\in\mathbb{R}^n$. In a pricing context, the vector ξ represents model parameters.

Example 5.1. (Fixed-grid local volatility) Consider a local volatility model described by the following SDE:

$$\forall t>0,\, \mathrm{d}S_t = r\,S_t\,\mathrm{d}t + \sigma(t,\log(S_t))\,S_t\,\mathrm{d}B_t$$

where $r\in\mathbb{R}$, B is a standard Brownian motion and for all $t\in[t^{(i)},t^{(i+1)}]$ and $s\in[s^{(j)},s^{(j+1)}]$:

$$\begin{aligned}\sigma(t,s) &= \sigma_{i,j} + \frac{s-s^{(j)}}{s^{(j+1)}-s^{(j)}}\left(\sigma_{i,j+1}-\sigma_{i,j}\right)\\ &\quad + \frac{t-t^{(i)}}{t^{(i+1)}-t^{(i)}}\left(\sigma_{i+1,j}-\sigma_{i,j}+\frac{s-s^{(j)}}{s^{(j+1)}-s^{(j)}}\left(\sigma_{i+1,j+1}-\sigma_{i,j+1}-\sigma_{i+1,j}+\sigma_{i,j}\right)\right)\end{aligned}$$

and for all $t>0$ and $s\in\mathbb{R}$, $\sigma(t,s)=\sigma(\tau(t),\chi(s))$, where $\tau(t)=\begin{cases} t & \text{if } \exists k\colon t\in[t^{(k)},t^{(k+1)}]\\ t^{(m)} & \text{if } t\geq t^{(m)}\\ t^{(0)} & \text{if } t<t^{(0)}\end{cases}$

and $\chi(s)=\begin{cases} s & \text{if } \exists k\colon s\in[s_k,s_{k+1}]\\ s^{(M)} & \text{if } s\geq s^{(M)}\\ s^{(0)} & \text{if } s<s^{(0)}\end{cases}$, and $0<t^{(0)}<\cdots<t^{(m)}$, $s^{(0)}<\cdots<s^{(M)}$ and $\sigma_{i,j}>0$ for all $i\in\{1,\ldots,m\}$ and $j\in\{1,\ldots,M\}$. For consistency of the interpolation formula above at the extremes, $t^{(m+1)}$ and $s^{(M+1)}$ will be taken to be any values respectively different from $t^{(m)}$ and $s^{(M)}$, their values having no impact on the value of $\sigma(t,s)$. We can again recast this model into the general form with the following mapping, with σ by construction also a function of ξ:

$$\begin{cases} \xi & := (r,\sigma_{0,0},\ldots,\sigma_{i,j},\ldots\sigma_{m,M})^\top\\ X_t^\xi & := S_t\\ f(t,X_t^\xi;\xi) & := r\,S_t\end{cases},\quad g(t,X_t^\xi;\xi) := \sigma(t,\log(X_t^\xi))\,X_t^\xi$$

□

We assume in what follows that the calibration instruments are vanilla European calls. Let Ξ, K, T be $\mathcal{F}_0$-measurable random variables supported on respectively $\mathbb{R}^n$, $\mathbb{R}_+$ and $\mathbb{R}_+^\star$ such that K and T are mutually independent and independent of Ξ. We are interested in pricing, conditional on random model parameters Ξ, random strike K and random maturity T, a product paying $Z^\Xi=(\psi(X_T^\Xi)-K)^+$ where $\psi\colon\mathbb{R}^d\to\mathbb{R}$ is piece-wise continuously differentiable and Lipschitz continuous. If we consider for simplicity[5.1] a numéraire $(e^{rt})_{t\geq 0}$ with r an $\mathcal{F}_0$-measurable risk-free rate which we will assume to be the first component of Ξ, then the price is given by $\mathbb{E}[e^{-rT}Z^\Xi|\Xi,K,T]$ and satisfies:

$$\mathbb{E}[e^{-rT}Z^\Xi|\Xi,K,T]=\varphi^\star(\Xi,K,T)$$

5.1. This does not limit the generality of the method in any way. Numéraires with stochastic interest rates can also be used without any issues.

with

$$\varphi^\star \in \underset{\varphi\in\mathcal{B}}{\operatorname{argmin}}\mathbb{E}[(\varphi(\Xi,K,T)-e^{-rT}Z^\Xi)^2] \tag{5.2}$$

where $\mathcal{B}$ is the space of Borel functions $\varphi\colon \mathbb{R}^{n+2}\to\mathbb{R}$ such that $\varphi(\Xi,K,T)$ is square integrable. Here we exploited the characterization of a conditional expectation as an L^2 projection. Indeed, we have for any such φ:

$$\begin{aligned}\mathbb{E}[(\varphi(\Xi,K,T)-e^{-rT}Z^\Xi)^2] &= \mathbb{E}[(\varphi(\Xi,K,T)-\mathbb{E}[e^{-rT}Z^\Xi|\Xi,K,T])^2]\\ &\quad+\underbrace{\mathbb{E}[\operatorname{var}(e^{-rT}Z^\Xi|\Xi,K,T)]}_{\text{independent of }\varphi}\end{aligned}$$

We propose to restrict the search space in (5.2) to the space of neural networks with a given architecture.

Definition 5.2. *Let α be a non-affine continuously differentiable activation function, applied element-wise, and let $m,p,q\in\mathbb{N}^\star$. Define $\mathcal{NN}_{m,p,q,\alpha}$ to be the set of functions $\mathbb{R}^{n+2}\ni x\mapsto\phi_{p+1}(x;W,b)$ such that $W_1\in\mathbb{R}^{m\times(n+2)}$, $W_2,\dots,W_p\in\mathbb{R}^{m\times m}$, $W_{p+1}\in\mathbb{R}^{q\times m}$, $b_1\dots,b_p\in\mathbb{R}^m$, $b_{p+1}\in\mathbb{R}^q$ and:*

$$\begin{aligned}\phi_0(x;W,b) &= x\\ \phi_i(x;W,b) &= \alpha(W_i\,\phi_{i-1}(x;W,b)+b_i),\forall i\in\{1,\dots,p\}\\ \phi_{p+1}(x;W,b) &= W_{p+1}\,\phi_p(x;W,b)+b_{p+1}\end{aligned} \tag{5.3}$$

$\mathcal{NN}_{m,p,q,\alpha}$ is then called the set of neural networks with p hidden layers, m neurons per hidden layer, $n+2$ inputs and q outputs, and α as its activation function.

Let $\mathcal{N}$ be such a set with $q=1$ (*i.e.* only one output neuron). An approach to approximate the pricing function $\varphi^\star$ would then be to find $\tilde\varphi$ such that:

$$\tilde\varphi\in\underset{\varphi\in\mathcal{N}}{\operatorname{argmin}}\mathbb{E}[(\varphi(\Xi,K,T)-e^{-rT}Z^\Xi)^2]$$

where the optimization becomes parametric as the neural networks in $\mathcal{N}$ are parameterized by their weights and biases, and the optimization can be done using vanilla stochastic gradient descent (SGD) or more elaborate accelerated SGD optimizers like ADAM [Kingma and Ba, 2014]. The approximation of $\mathbb{E}[e^{-rT}Z^\Xi|\Xi,K,T]$ would then be $\tilde\varphi(\Xi,K,T)$.

The restriction to neural networks is mainly motivated by the following density result:

Theorem 5.3. (Universal Approximation Theorem for Deep Narrow Networks[Kidger and Lyons, 2020]) *Let α be a non-affine continuously differentiable activation function, $q\in\mathbb{N}^\star$ and let $K\subset\mathbb{R}^{n+2}$ be compact. Then $\bigcup_{p\in\mathbb{N}^\star}\mathcal{NN}_{q+n+4,p,q,\alpha}$ is dense in $\mathcal{C}(K,\mathbb{R}^q)$ with respect to the topology of uniform convergence.*

5.3 Regularizing with Sobolev Training

The learning task however will suffer from increased variance compared to price interpolation approaches and stochastic gradient descent will yield noisy updates because of gradient estimators having high variances. The high variance can be overcome by producing a large enough data-set and using large batch sizes during the SGD iterations. In addition to the potential variance problem, regularization may be needed when considering large neural networks to avoid over-fitting payoffs. This is usually [LeCun et al., 2015] done by imposing generic restrictions on the neural network weights such as L^1 or L^2 regularization.

In [Huge and Savine, 2020], the authors propose to tackle both issues by instead adding a *regularizing* term which penalizes the error when projecting the path-wise derivatives of the payoff, which will have the effect of both artificially increasing the size of the data-set (while profiting from the common computations for the path-wise derivatives) and regularizing the learning procedure by adding constraints on the behavior of the network locally around sample points. Although the authors focused on randomizing only the initial value of the SDE and not its parameters, when recast in our calibration setting it leads to having to solve the following learning problem:

$$\tilde{\tilde{\varphi}} \in \underset{\varphi\in\mathcal{N}}{\operatorname{argmin}} \mathbb{E}[(\varphi(\Xi,K,T) - e^{-rT}Z^{\Xi})^2] + \sum_{k=1}^{n+2} \lambda_k \mathbb{E}[(\partial_k\varphi(\Xi,K,T) - \partial_k(e^{-rT}Z^{\Xi}))^2] \tag{5.4}$$

where $\lambda \in (\mathbb{R}_+^\star)^{n+2}$ and ∂_k is the partial derivative with respect to the k-th component of the concatenation of Ξ and (K,T). More precisely:

Definition 5.4. (Point-wise gradient of a parameterized random variable) *If $Y(\xi)$ is a real valued random variable that is parameterized by $\xi \in \mathbb{R}^n$ then we define the point-wise gradient $\nabla Y(\xi)$ of $Y(\xi)$ with respect to ξ as $\nabla Y(\xi) = (\partial_1 Y(\xi), \dots, \partial_n Y(\xi))^\top$ where for every $i \in \{1,\dots,n\}$ we have:*

$$\partial_i Y(\xi) := \lim_{\varepsilon\to 0} \frac{1}{\varepsilon}(Y(\xi + \varepsilon\, e_i) - Y(\xi))$$

with $(e_1,\dots,e_n)$ being the canonical basis in $\mathbb{R}^n$, where the limit is taken point-wise over Ω.

We give below sufficient assumptions in our randomized pricing framework so that these point-wise (or *path-wise* in our case) derivatives exist and are unbiased estimators of the respective derivatives of the price. We refer the reader to [Broadie and Glasserman, 1996; Glasserman, 2004] for a more general and comprehensive treatment of path-wise sensitivities.

Assumption 5.5. $\lim_{\varepsilon\to 0} \frac{1}{\varepsilon}(X_T^{\xi+\varepsilon e_i} - X_T^{\xi})$ *exists with probability 1 for all $i \in \{1,\dots,n\}$ and $\xi \in \mathbb{R}^n$.*

Assumption 5.6. $\mathbb{Q}(\psi(X_T^{\xi}) = K) = 0$ *for all $\xi \in \mathbb{R}^n$.*

Assumption 5.7. *There exists $\kappa > 0$ such that for all $\xi_1, \xi_2 \in \mathbb{R}^n$ we have*

$$\mathbb{E}[\|X_T^{\xi_1} - X_T^{\xi_2}\|] \le \kappa\, \|\xi_1 - \xi_2\|$$

Assumptions 5.5 and 5.6 in particular imply that the point-wise gradient of the call payoff with respect to model parameters exists with probability 1 and justify why it is enough to assume piece-wise differentiability. Assumption 5.7[5.2] along with the Lipschitz regularity of the positive part and of ψ implies that the call payoff is Lipschitz continuous with respect to the model parameters. All these assumptions together with the Vitali convergence theorem ensure the existence of the point-wise gradients with probability 1 and that we can interchange gradients and expectations, *i.e.*

$$\nabla\mathbb{E}[e^{-rT}Z^{\Xi}|\Xi,K,T] = \mathbb{E}[\nabla(e^{-rT}Z^{\Xi})|\Xi,K,T] \tag{5.5}$$

5.2. We have chosen to express Assumption 5.7 such that we can use the Vitali convergence theorem, this is in contrast with the choice in [Broadie and Glasserman, 1996; Glasserman, 2004] to write the assumption in an almost-sure manner which allows the use of the dominated convergence theorem but makes it hard to verify it in very general pricing models. Assumption 5.7 can easily be verified by first bounding $\mathbb{E}[\|X_T^{\xi_1} - X_T^{\xi_2}\|^2]$ using Itô isometry and Grönwall's lemma.

The extension to the differentiation with respect to the strike and the maturity is trivial. Note that in discrete-time the payoff will be by construction not differentiable with respect to the maturity. However, one can circumvent this issue by deducing the partial derivative of the price with respect to the maturity using spatial derivatives thanks to the Fokker-Planck equation (see Appendix 5.A).

Hence, the learning problem statement in (5.4) explicitly seeks to orthogonally project not only payoffs but also their path-wise sensitivities, motivated by (5.5), which is consistent with our calibration context as a desired feature in the price approximation is access to accurate gradients so that one can use gradient-based optimizers during the calibration phase.

Example 5.8 shows how path-wise derivatives can be computed in a time-discretized[5.3] version of our model in Example 5.1; one can notice in particular the amount of reusable common sub-expressions which make path-wise derivative calculations cheaper than simulating new trajectories of payoffs.

Example 5.8. (Path-wise sensitivities in a fixed-grid local volatility model) Assume an Euler-Maruyama scheme for the log-dynamics of the model described in Example 5.1:

$$\hat{s}_{i+1} = \hat{s}_i + \left(r - \frac{1}{2}\sigma(t_i, \hat{s}_i)^2 \right) h + \sigma(t_i, \hat{s}_i)\,\sqrt{h}\,G_{i+1} \quad , \forall i \in \{0, \dots, I-1\}$$

for some time-grid $0 = t_0 < \dots < t_i = i\,h < \dots < t_I = T$ with constant step size $h > 0$, a sequence of independent standard Gaussian variables $(G_i)_{i \geq 1}$, and random $\mathcal{F}_0$-measurable local volatility nodes $(\sigma_{i,j})_{\substack{1 \leq i \leq m \\ 1 \leq j \leq M}}$ and random $\mathcal{F}_0$-measurable risk-free rate r. The discrete process $(\hat{s}_i)_{0 \leq i \leq I}$ is then a time-discretized approximation of $(\log(S_t))_{t \geq 0}$ at the discrete time steps $t_0, \dots, t_I$. Let $i \in \{0, \dots, I-1\}$. By differentiating both the LHS and RHS in the discretized equation above and setting $\alpha_{i+1} = \sqrt{h}\,(G_{i+1} - \sigma(t_i, \hat{s}_i)\,\sqrt{h})\,\partial_s\sigma(t_i, \hat{s}_i)$, one can write[5.4]:

$$\begin{aligned} \partial_r \hat{s}_{i+1} &= \partial_r \hat{s}_i + h + \alpha_{i+1}\,\partial_r \hat{s}_i \\ \partial_{\sigma_{u,v}} \hat{s}_{i+1} &= \partial_{\sigma_{u,v}} \hat{s}_i + \alpha_{i+1}\,\partial_{\sigma_{u,v}} \hat{s}_i \end{aligned}$$

for all $1 \leq u \leq m$ and $1 \leq v \leq M$. It then remains to evaluate $\partial_s\sigma(t_i, \hat{s}_i)$, for which we defer the calculations to Appendix 5.B.
□

This class of learning problems, where one is given not only values of the function to be approximated (or, in our case, the random variable to be projected) but also partial derivatives was, to our knowledge, first formulated by [Czarnecki et al., 2017] who coined the name *Sobolev training* and has been ubiquitous in Machine Learning research involving physical phenomena [Son et al., 2021; Vlassis and Sun, 2021], where the learning is augmented by either values of the partial derivatives, or relationships between partial derivatives using domain-specific PDEs.

5.3. The assumptions 5.5, 5.6 and 5.7 also apply to the discrete case and the notion of path-wise differentiation and the unbiasedness (*i.e.* Equation (5.5)) of the path-wise derivatives readily extend to time-discretized models.

5.4. This approach to computing derivatives by iterating through time in a forward way is more attractive when considering an implementation on GPUs. Indeed, at any time-step, all the calculations in Example 5.8 are done locally and registers and GPU caching can be used efficiently. This is in opposition to the traditional backward AAD where one would have to write to the global memory at each time-step and then later read from those same memory locations again which severely hurts performance in GPU implementations.

5.4 Complex-step Sobolev Training

5.4.1 Restricting to Stochastic Directional Derivatives

Evaluating the partial derivatives of the neural network in (5.4) can be a time-consuming process depending on how the computation is done. We give two classical approaches with which those derivatives can be computed exactly. The calculations outlined here form the backbone of modern *algorithmic differentiation* and we refer the reader to [Baydin et al., 2018] for a more comprehensive survey and more technical discussions and to [Savine, 2018] for a reference in the context of computing sensitivities of the price of financial derivatives.

Assume $\mathcal{N} = \mathcal{NN}_{m,p,1,\alpha}$ for some $m, p \in \mathbb{N}^\star$ and a non-affine twice continuously differentiable activation function α and let $\varphi \in \mathcal{N}$, *i.e.* there exist $W_1 \in \mathbb{R}^{m\times(n+2)}$, $W_2, \dots, W_p \in \mathbb{R}^{m\times m}$, $W_{p+1} \in \mathbb{R}^{1\times m}$, $b_1 \dots, b_p \in \mathbb{R}^m$, $b_{p+1} \in \mathbb{R}$ such that $\varphi(x) = \phi_{p+1}(x; W, b)$ for all $x \in \mathbb{R}^{n+2}$. Denoting by $J_\cdot$ and ∇ the jacobian matrix and gradient with respect to x and by α' the derivative of the activation function α (applied element-wise), we have:

$$\begin{aligned} J_{\phi_0}(x; W, b) &= I_{n+2} \\ J_{\phi_i}(x; W, b) &= \operatorname{diag}(\alpha'(W_i\, \phi_{i-1}(x; W, b) + b_i))\, W_i\, J_{\phi_{i-1}}(x; W, b), \forall i \in \{1, \dots, p\} \\ \nabla\phi_{p+1}(x; W, b) &= J_{\phi_p}(x; W, b)^\top W_{p+1}^\top \end{aligned} \tag{5.6}$$

This naturally defines the so-called *forward* approach to compute the gradient $\nabla\phi_{p+1}(x; W, b)$. Indeed one starts with $J_{\phi_0}(x; W, b)$ and recursively computes $J_{\phi_i}(x; W, b)$ given $J_{\phi_{i-1}}(x; W, b)$ for all $i \in \{1, \dots, p\}$ to deduce $\nabla\phi_{p+1}(x; W, b)$ at the end, effectively performing the product of jacobian matrices from left to right.

Another approach would be to introduce $\delta_0, \dots, \delta_p$ as follows:

$$\begin{aligned} \delta_0(x; W, b) &= W_{p+1} \\ \delta_i(x; W, b) &= \delta_{i-1}(x; W, b) \operatorname{diag}(\alpha'(W_{p-i+1}\, \phi_{p-i}(x; W, b) + b_{p-i+1}))\, W_{p-i+1}, \\ &\qquad \forall i \in \{1, \dots, p\} \\ \nabla\phi_{p+1}(x; W, b) &= \delta_p(x; W, b)^\top \end{aligned} \tag{5.7}$$

This defines a *backward*, or *reverse*, approach to compute the gradient $\nabla\phi_{p+1}(x; W, b)$, since we are in effect doing a product of jacobian matrices from right to left.

The major difference between both approaches lies in the fact that the backward approach consists, at least in theory, of less arithmetic operations for the same result as the forward approach. Indeed, by counting in terms of m, n and p the additions and multiplications involved in the matrix operations[5.5] and the evaluations of the derivative of the activation function and discounting the evaluations of pre-activations[5.6], one can show that the forward approach performs $\Theta(m^2\, n\, p)$ operations[5.7] while the backward approach performs only $\Theta(m^2\, p)$ operations. Indeed, during the forward approach, one iteratively computes products of full jacobian matrices and more precisely matrix-matrix products, while in the backward approach one computes only vector-matrix products given that the neural network has only one output neuron.

5.5. One should pay attention to the fact that the multiplication of an arbitrary square matrix A with a diagonal matrix B on the left (resp. on the right), *i.e.* $A\, B$ (resp. $B\, A$), should be implemented by multiplying the i-th column (resp. row) of A by the i-th entry on the diagonal of B instead of using generic matrix multiplication. This is taken into account in the complexity calculations.

5.6. *i.e.* the calculation of the terms $W_i\, \phi_{i-1}(x; W, b) + b_i$ which are assumed to have already been computed during a forward pass to compute the neural network's outputs.

5.7. Given two real-valued sequences $(u_n)_n$ and $(v_n)_n$, we say that $u_n = \Theta(v_n)$ if $u_n = \mathcal{O}(v_n)$ and $v_n = \mathcal{O}(u_n)$.

However, this complexity analysis neglects the memory occupation and time spent during loads and stores from and in memory that are necessary for the backward phase as one has to store the pre-activations of each layer during the forward phase in order to later load them and use them during the backward computation. The iterates $\delta_0, \dots, \delta_p$ in the backward computations are also stored in memory so that another backward differentiation with respect to the weights of the neural network (and not its inputs), more commonly called *back-propagation*, can be performed in order to be able to make an SGD step towards solving (5.4). This second layer of backward differentiation is done automatically by all major neural network libraries using *adjoint algorithmic differentiation* (AAD) [Abadi et al., 2016; Bradbury et al., 2018; Paszke et al., 2019]. This memory cost, both in occupied space and in access times, is further exacerbated by the need to compute the gradient $\nabla\phi_{p+1}(x; W, b)$ separately for each point x in the sample when performing empirical risk minimization.

In [Czarnecki et al., 2017], the authors point out that it is possible to reformulate the loss function in such a way that one would need only directional derivatives instead of full gradients in order to address the previous computational issue. We propose a more general result in the same spirit.

Proposition 5.9. *Let u be an L^2-integrable random vector supported on $\mathbb{R}^{n+2}$ with zero-mean components such that $\mathrm{cov}(u) = \mathrm{diag}(\lambda_1, \dots, \lambda_{n+2})$ and assume that u is independent of Ξ, K, T, Z^Ξ. We have:*

$$\mathbb{E}[(u^\top \nabla\varphi(\Xi, K, T) - u^\top \nabla(e^{-rT} Z^\Xi))^2] = \sum_{k=1}^{n+2} \lambda_k \, \mathbb{E}[(\partial_k\varphi(\Xi, K, T) - \partial_k(e^{-rT} Z^\Xi))^2] \tag{5.8}$$

Proposition 5.9 suggests that one can perform stochastic gradient descent to solve (5.4) by computing only one directional derivative along a random direction instead of having to compute full gradients during each SGD iteration. The authors in [Czarnecki et al., 2017] suggested to draw u from a uniform distribution on the unit sphere but no discussion of the motivation behind this choice was provided. We give the optimal distribution among those verifying the assumptions of Proposition 5.9 when seeking to minimize the additional variance created by randomizing the direction of differentiation:

Proposition 5.10. *Denote $\ell := \nabla\varphi(\Xi, K, T) - \nabla(e^{-rT} Z^\Xi)$ and $\mathcal{Z} = \sigma(\Xi, K, T, Z^\Xi)$. Under the assumptions of Proposition 5.9, u minimizes $\mathbb{E}[\mathrm{var}((u^\top \ell)^2 | \mathcal{Z})]$ iff:*

$$u \sim \left(\sqrt{\lambda_1}\, R_1, \dots, \sqrt{\lambda_{n+2}}\, R_{n+2}\right)$$

where $R_1, \dots, R_N$ are i.i.d Rademacher variables, i.e. $\mathbb{Q}(R_i = -1) = \mathbb{Q}(R_i = 1) = \frac{1}{2}$.

Proof. We have:

$$\begin{aligned} \mathrm{var}((u^\top \ell)^2 | \mathcal{Z}) &= \mathbb{E}[(\ell^\top (u\, u^\top - \mathrm{diag}(\lambda))\, \ell)^2 | \mathcal{Z}] \\ &= \mathbb{E}\left[\left(\sum_{i=1}^{n+2} \ell_i^2 \, (u_i^2 - \lambda_i) + 2 \sum_{1 \le i < j \le n+2} \ell_i \ell_j u_i u_j \right)^2 \middle| \mathcal{Z} \right] \end{aligned}$$

Denote $A := \sum_{i=1}^{n+2} \ell_i^2 \, (u_i^2 - \lambda_i)$ and $B := \sum_{1 \le i < j \le n+2} \ell_i \ell_j u_i u_j$. We then have:

$$\mathrm{var}((u^\top \ell)^2 | \mathcal{Z}) = \mathbb{E}[A^2 + 4\, B^2 + 4\, A\, B | \mathcal{Z}]$$

and

$$\begin{aligned} A^2 &= \sum_{i=1}^{n+2} \ell_i^4 \,(u_i^2-\lambda_i)^2 + 2 \sum_{1\le i<j\le n+2} \ell_i^2\,\ell_j^2\,(u_i^2-\lambda_i)(u_j^2-\lambda_j) \\ B^2 &= \sum_{1\le i<j\le n+2} \ell_i^2\,\ell_j^2\,u_i^2\,u_j^2 + \sum_{\substack{1\le i<j\le n+2\\ 1\le \imath<\jmath\le n+2\\ (i,j)\neq(\imath,\jmath)}} \ell_i\,\ell_\imath\,\ell_j\,\ell_\jmath\,u_i\,u_\imath\,u_j\,u_\jmath \\ A\,B &= \sum_{r=1}^{n+2} \sum_{1\le i<j\le n+2} \ell_r^2\,\ell_i\,\ell_j\,(u_r^2-\lambda_r)\,u_i\,u_j \end{aligned}$$

Notice that whenever $i \neq j$, we have:

$$\begin{aligned} \mathbb{E}[\ell_i^2\,\ell_j^2\,(u_i^2-\lambda_i)(u_j^2-\lambda_j)|\mathcal{Z}] &= \ell_i^2\,\ell_j^2\,\mathbb{E}[u_i^2-\lambda_i]\,\mathbb{E}[u_j^2-\lambda_j] = 0 \\ \mathbb{E}[\ell_i^2\,\ell_j^2\,u_i^2\,u_j^2|\mathcal{Z}] &= \ell_i^2\,\ell_j^2\,\lambda_i\,\lambda_j \end{aligned}$$

Let $i, \imath, j, \jmath$ be integers such that $1 \le i < j \le n+2$ and $1 \le \imath < \jmath \le n+2$ and $(i,j) \neq (\imath,\jmath)$. We have the following:

- if $i = \imath$ and $j \neq \jmath$: $\mathbb{E}[\ell_i\,\ell_\imath\,\ell_j\,\ell_\jmath\,u_i\,u_\imath\,u_j\,u_\jmath|\mathcal{Z}] = \ell_i^2\,\ell_j\,\ell_\jmath\,\mathbb{E}[u_i^2]\,\mathbb{E}[u_j]\,\mathbb{E}[u_\jmath] = 0$
- if $i \neq \imath$ and $j = \jmath$: $\mathbb{E}[\ell_i\,\ell_\imath\,\ell_j\,\ell_\jmath\,u_i\,u_\imath\,u_j\,u_\jmath|\mathcal{Z}] = \ell_i\,\ell_\imath\,\ell_j^2\,\mathbb{E}[u_i]\,\mathbb{E}[u_\imath]\,\mathbb{E}[u_j^2] = 0$

Let r be an integer such that $1 \le r \le n+2$. We can distinguish the following cases:

- if $r \neq i$ and $r \neq j$: $\mathbb{E}[\ell_r^2\,\ell_i\,\ell_j\,(u_r^2-\lambda_r)\,u_i\,u_j|\mathcal{Z}] = \ell_r^2\,\ell_i\,\ell_j\,\mathbb{E}[u_r^2-\lambda_r]\,\mathbb{E}[u_i]\,\mathbb{E}[u_j] = 0$
- if $r = i$: $\mathbb{E}[\ell_r^2\,\ell_i\,\ell_j\,(u_r^2-\lambda_r)\,u_i\,u_j|\mathcal{Z}] = \ell_r^3\,\ell_j\,\mathbb{E}[(u_r^2-\lambda_r)\,u_r]\,\mathbb{E}[u_j] = 0$
- if $r = j$: $\mathbb{E}[\ell_r^2\,\ell_i\,\ell_j\,(u_r^2-\lambda_r)\,u_i\,u_j|\mathcal{Z}] = \ell_r^3\,\ell_i\,\mathbb{E}[(u_r^2-\lambda_r)\,u_r]\,\mathbb{E}[u_i] = 0$

Hence,

$$\operatorname{var}((u^\top \ell)^2|\mathcal{Z}) = \sum_{i=1}^{n+2} \ell_i^4\,\mathbb{E}[(u_i^2-\lambda_i)^2] + 4 \sum_{1\le i<j\le n+2} \ell_i^2\,\ell_j^2\,\lambda_i\,\lambda_j \tag{5.9}$$

which is minimized iff $u_i^2 = \lambda_i$ a.s. for all $i \in \{1,\dots,n+2\}$. Assume this is verified. Since u is centered, we have $\mathbb{Q}(u_i = \sqrt{\lambda_i}) = \mathbb{Q}(u_i = -\sqrt{\lambda_i}) = \frac{1}{2}$ for every $i \in \{1,\dots,n+2\}$ and this concludes the proof. □

Remark 5.11. The expectation of the conditional variance in (5.9) also happens to be the additional variance caused by the introduction of a random direction of differentiation. Indeed, reusing the notation of Proposition 5.10 and setting $\tilde{\ell} := \varphi(\Xi, K, T) - e^{-rT} Z^\Xi$, we have:

$$\operatorname{var}(\tilde{\ell}^2 + (u^\top \ell)^2) = \operatorname{var}(\tilde{\ell}^2) + \operatorname{var}((u^\top \ell)^2) + 2\operatorname{cov}(\tilde{\ell}^2, (u^\top \ell)^2)$$

Using the tower property one can show that:

$$\operatorname{cov}(\tilde{\ell}^2, (u^\top \ell)^2) = \operatorname{cov}(\tilde{\ell}^2, \mathbb{E}[(u^\top \ell)^2|\mathcal{Z}])$$

We also have from the total variance formula that:

$$\operatorname{var}((u^\top \ell)^2) = \mathbb{E}[\operatorname{var}((u^\top \ell)^2|\mathcal{Z})] + \operatorname{var}(\mathbb{E}[(u^\top \ell)^2|\mathcal{Z}])$$

Hence:

$$\mathrm{var}(\tilde{\ell}^2 + (u^\top \ell)^2) = \mathrm{var}(\tilde{\ell}^2 + \mathbb{E}[(u^\top \ell)^2|\mathcal{Z}]) + \mathbb{E}[\mathrm{var}((u^\top \ell)^2|\mathcal{Z})]$$

and the conclusion is immediate by noticing that:

$$\tilde{\ell}^2 + \mathbb{E}[(u^\top \ell)^2|\mathcal{Z}] = (\varphi(\Xi, K, T) - e^{-rT} Z^\Xi)^2 + \sum_{k=1}^{n+2} \lambda_k \left(\partial_k \varphi(\Xi, K, T) - \partial_k (e^{-rT} Z^\Xi)\right)^2$$

Hence, the computational gain provided by differentiating along random directions instead of computing full gradients comes at the cost of a necessarily higher variance. In this regard, Proposition 5.10 helps reduce this additional variance as much as possible by judiciously choosing the distribution of u. By choosing the optimal distribution, the added variance is then:

$$\mathbb{E}[\mathrm{var}((u^\top \ell)^2|\mathcal{Z})] = 4 \sum_{1 \le i < j \le n+2} \lambda_i \lambda_j \mathbb{E}[\ell_i^2 \ell_j^2]$$

□

One can then compute the directional derivatives using a forward approach, yielding similar complexity to the backward approach in terms of the arithmetic operations, while performing less memory accesses as no jacobians need to be stored in the forward approach. Indeed, one can iterate jointly on the forward approach and the calculation of the neural network output and share pre-activation values between both computations. Although the same directional derivatives computation can be done using the backward approach, even in terms of solely the arithmetic operations it is sub-optimal to do so. Figure 5.1 shows that when the number of inputs is greater than the number of outputs (*i.e.* $d_0 > d_2$ in the figure), vector-matrix multiplications from left to right perform less work and are thus faster.

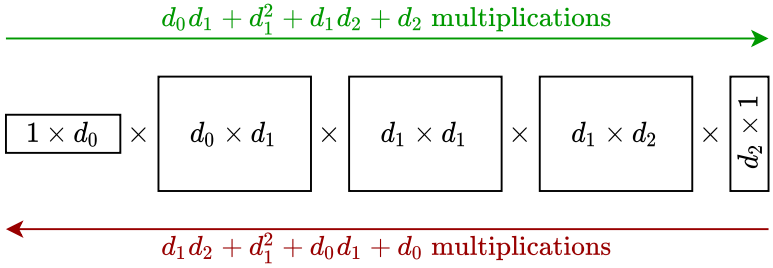

Figure 5.1. Jacobian multiplications when differentiating a neural network with 3 hidden layers having d_0 inputs, d_1 neurons per hidden layer and d_2 outputs, with respect to its inputs along a fixed direction.

5.4.2 Faster Directional Derivatives with Complex-step Differentiation

Notice that for a given direction u, the directional derivative $u^\top \nabla \varphi(\Xi, K, T)$ can also be approximated using a finite difference along the direction u at the cost of two[5.8] evaluations for both the output of the network and its directional derivative and would have to perform

5.8. or three when using a centered finite difference approximation.

less work than an exact directional derivative with the forward approach given that the latter will have the additional overhead of the diagonal matrix multiplication in (5.6), which has quadratic complexity in the number of hidden units, while the forward pass using (5.3) doesn't.

However, the finite difference method suffers from round-off errors when implemented using finite precision arithmetic because of the substraction involved. In particular, the approximation can become unstable as shown in the example of Figure 5.2 and can fail for step sizes that are less than $\sim\sqrt{\epsilon}$ ($\sim\sqrt[3]{\epsilon}$ if using the central finite difference method) where ϵ is the machine epsilon [Sauer, 2011]. This is further exacerbated by the fact that computations on the GPU are preferably done in single precision (where the machine epsilon is of the order of 10^{-7}) as switching to double precision (the machine epsilon becoming then $\sim 10^{-16}$) comes with a performance penalty. This greatly limits the degree to which one can reduce the approximation error by reducing the step size if one wants to preserve the computational advantage[5.9] of single precision, especially on GPUs.

In [Martins et al., 2003; Squire and Trapp, 1998], it is shown that if the function that is being differentiated admits an analytic extension, then these numerical issues can be overcome using a so-called *complex-step differentiation* instead of a finite difference approximation. We first present the result in scalar form, which can easily be verified using a Taylor expansion for holomorphic functions (we refer the reader to [Lang, 2003] for a classic treatment of complex analysis):

Proposition 5.12. (Complex-step differentiation) *Let $x \in \mathbb{R}$ and let $F : \mathbb{R} \to \mathbb{R}$ be thrice differentiable on a neighborhood V of x and assume that F can be extended analytically on V. By identifying F with its analytic extension on V, we have:*

$$\frac{1}{\varepsilon}\operatorname{Im}(F(x + \mathrm{i}\,\varepsilon)) = F'(x) + \mathcal{O}(\varepsilon^2)$$

where i *is the imaginary unit and* Im *is the imaginary part operator.*

Notice that the approximation in Proposition 5.12 does not involve a difference, hence being less subject to the round-off errors encountered in finite difference implementations, and is of order two, having thus the same order as a central finite difference but with better numerical stability. In practice, the complex step size can be taken to be as small as the machine epsilon, yielding an approximation that is almost indistinguishable from the exact value of the derivative in finite precision. In the example of Figure 5.2, the absolute error is of the order of the machine epsilon when ε is sufficiently small. Switching to holomorphic functions on $\mathbb{C}^{n+2}$, and assuming that the neural network φ admits an analytic extension with which we identify it, we can finally write:

$$u^\top \nabla \varphi(x) = \frac{1}{\varepsilon}\operatorname{Im}(\varphi(x + \mathrm{i}\,\varepsilon\, u)) + \mathcal{O}(\varepsilon^2)$$

5.9. Not only are double-precision arithmetics, in theory, twice as slow as single-precision arithmetics, recent NVidia GPUs give even more advantage to single-precision (albeit in a specialized TensorFloat32 format) with speedups up to $\times 8$ with respect to regular single-precision (*i.e.* floats or FP32) on the A100 GPU thanks to *Tensor cores*, at least according to NVidia's official specifications [NVIDIA Corporation, 2020a]. Of course, in addition to the computational speed, there is also the difference in memory storage, as double-precision will necessarily use twice as much memory as single-precision, and also involve twice as many memory exchanges which can be penalizing in situations that are memory-bound.

where Im is applied element-wise. Notice that the output of the neural network can be deduced immediately without performing any additional evaluation, with the same order of error:

$$\varphi(x) = \mathrm{Re}(\varphi(x + \mathrm{i}\,\varepsilon\,u)) + \mathcal{O}(\varepsilon^2)$$

with Re being the real part operator applied element-wise. Hence both values and directional derivatives can be computed at the same time with low approximation error.

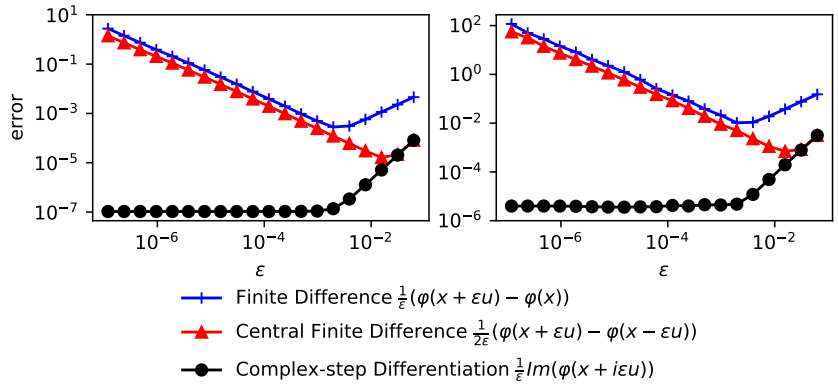

Figure 5.2. Sample average of the absolute value of absolute (**left**) and relative (**right**) errors, when approximating the directional derivative of a randomly initialized neural network φ, with 28 inputs, 6 hidden layers, 112 hidden units per layer and a Softplus activation, with respect to its inputs, using each of the finite difference, central finite difference and complex-step differentiation methods. The average is done over an i.i.d sample of $2^{14} = 16384$ errors each corresponding to a network input vector x with components drawn independently from $\mathcal{U}([-\sqrt{3}, \sqrt{3}])$ and a direction u drawn as in Proposition 5.10 with $\lambda_1 = \cdots = \lambda_{28} = 1$. Plots are in log-log scale.

In practice, a neural network can be rendered analytic by choosing activation functions that can be extended analytically, since the affine layers admit trivial analytic extensions.

Example 5.13. (Analytic Softplus activation) Consider the Softplus activation function:

$$\alpha(x) = \log(1 + \exp(x)), \quad \forall x \in \mathbb{R}$$

and which is applied element-wise when supplied with a vector in its input. It can be extended naturally to a holomorphic function by extending log with $z \mapsto \log|z| + i\,\mathrm{Arg}(z)$ on $\mathbb{C} \setminus \{0\}$, with Arg being the principal argument on $(-\pi, \pi]$, and exp with $x + \mathrm{i}\,y \mapsto \exp(x)\,(\cos(y) + i\sin(y))$ on $\mathbb{C}$. Hence, except where undefined[5.10], we can use the following[5.11] extension for all $x, y \in \mathbb{R}$:

$$\alpha(x+\mathrm{i}y) = \frac{1}{2}\log(1+\exp(2x)+2\exp(x)\cos(y)) + \mathrm{i}\,\mathrm{atan2}(\exp(x)\sin(y), 1+\exp(x)\cos(y))$$

where $\mathrm{atan2}(y, x) = \mathrm{Arg}(x + \mathrm{i}\,y)$ and is implemented in most math libraries. One then extends α to vectors in $\mathbb{C}^{n+2}$ by applying it element-wise.

5.10. which is not an issue as the set of such points is of measure zero.

5.11. One still needs to treat overflow issues, caused by the exponential, based on the sign of x in the implementation.

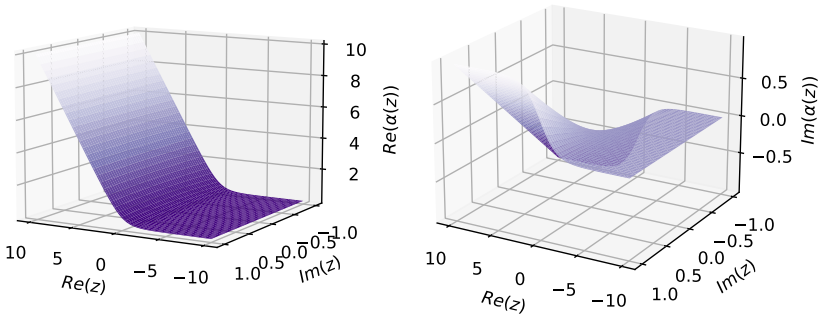

Figure 5.3. An analytic extension of the Softplus activation. **Left:** real part, **right:** imaginary part.

□

The main difficulty then resides in implementing support for complex-valued inputs for neural networks in such a way that the library's automatic differentiation with respect to the network weights remains possible. At the time of this writing, this is done in `pytorch` for instance using the notion of Wirtinger differentiation [Bouboulis, 2010] which is more general than holomorphic differentiability. However, we chose to specialize in holomorphic functions as these enjoy the Cauchy-Riemann property, which we recall below in the scalar case, as opposed to general Wirtinger differentiable functions.

Theorem 5.14. (Cauchy-Riemann equations) *Let $v\colon z = x + \mathrm{i}\, y \mapsto v_0(x, y) + \mathrm{i}\, v_1(x, y)$ be defined in a neighborhood V of $z_0 = x_0 + \mathrm{i}\, y_0$ and assume v is holomorphic on V. Then the partial derivatives $\partial_x v_0$, $\partial_y v_0$, $\partial_x v_1$ and $\partial_y v_1$ exist at (x_0, y_0) and we have:*

$$\begin{aligned} \partial_x v_0(x_0, y_0) &= \partial_y v_1(x_0, y_0) \\ \partial_y v_0(x_0, y_0) &= -\partial_x v_1(x_0, y_0) \end{aligned}$$

The Cauchy-Riemann equations in particular allow one to compute only two partial derivatives (*e.g.* the partial derivatives of the real part of the output with respect to the real and imaginary parts of the input) and deduce the other two partial derivatives with at most a change of sign. This in particular allows us to directly hard-code[5.12] the fact that one needs to store only two partial derivatives and use those to immediately get all four partial derivatives (of the real and imaginary parts of the output with respect to the real and imaginary parts of the inputs) at all intermediate layers during the back-propagation, while Wirtinger differentiation requires in principle keeping track of all four partial derivatives as the Cauchy-Riemann equations are not necessarily verified for a Wirtinger differentiable function.

Using our custom holomorphic implementation in `C++` yields speed-ups between $\times 1.4$ and $\times 2$ compared to `PyTorch`'s Wirtinger-based back-propagation[5.13] in our benchmarks of back-propagation times in 5.5.2. We also get speed-ups between $\times 3.6$ and $\times 7.6$ compared to an exact calculation of directional derivatives using the forward approach.

5.12. We do this using the `C++ libtorch` library by reimplementing the needed neural network blocks, such as activations and layers, by inheriting from the `torch::autograd::Function` class and exploiting the Cauchy-Riemann property in our custom `torch::autograd::Function::backward` implementation.

5.13. We used PyTorch's *just-in-time* (JIT) compilation mechanism when benchmarking the vanilla complex implementation on Python, which removes in practice most of the Python overhead. Hence the speed-up is mostly explained by our specialization to holomorphic functions and not merely by switching to `C++`. The switch to `C++` was needed only because our custom implementation using CUDA kernels could not, at the moment of this writing, be used with the JIT mechanism, thus severely penalizing it on Python.

5.5 Numerical Case-study: Fixed-grid Local Volatility

5.5.1 Setup of The Experiments

We consider the problem of fitting a 5×5 fixed-grid local volatility model, as described in Example 5.1, on observed quotes of call and put prices. We set $X_0 = 1$ and for the local volatility grid we use a time grid $\left\{\frac{1}{12} + \frac{(2-1/12)\,k}{4}\right\}_{0\le k\le 4}$ and a spatial grid $\left\{\log(0.25) + \frac{\log(2.1/0.25)\,k}{5}\right\}_{0\le k\le 4}$.

In this example then, we have $d = 1$, $\psi = \mathrm{Id}$ and $n = 26$ corresponding to 25 local volatility nodes and the risk-free rate.

We generate $2^{23} \approx 8$ million Monte Carlo samples of (Ξ, K, T, Z) assuming a time discretization with an Euler-Maruyama scheme. We sample each factor as in Table 5.1, which essentially amounts to seeking an approximation of the pricing function for parameters in the described ranges. Since the simulations are fast (see 5.5.2), we also use a modest variance reduction by replacing the payoffs with an average over 32 realizations[5.14] of the payoff conditional on the same parameter realization, since both have the same expectation conditional on (Ξ, K, T, Z).

Factor	Description	Distribution
r	risk-free rate	$\mathcal{U}([0, 0.05])$
K	strike price	$\mathcal{U}([0.25, 2.1])$
T	maturity	$\mathcal{U}([0.05, 2.5])$
$\sigma_{i,j}$	local volatility at node (i, j)	$\mathcal{U}([0.1, 2.0])$

Table 5.1. Distributions of product and model parameters.

We used a neural network with $p = 6$ layers, $m = 56$ hidden units per hidden layer, an analytic Softplus activation and, because we are specializing in call prices, $q = 2$ outputs constrained to be valued on $[0, 1]$, on which an inner-product with $(X_0, -e^{-rT}K)^\top$ is then performed to get an estimate of the call price. More precisely, let $\mathcal{N} = \mathcal{NN}_{56,6,2,\alpha}$ and denote by $[.]_k$ the k-th coordinate of its argument (with k zero-indexed). Define:

$$\mathcal{H} := \{\mathbb{R}^n \times \mathbb{R} \times \mathbb{R} \ni (\xi, k, \tau) \mapsto X_0\, s([\phi(\xi, k, \tau)]_0) - e^{-[\xi]_0 \tau} k\, s([\phi(\xi, k, \tau)]_1) \colon \phi \in \mathcal{N}\}$$

where $s \colon \mathbb{R} \ni x \mapsto \frac{1}{1+\exp(-x)}$. Then the minimization is carried over $\mathcal{H}$ instead of $\mathcal{N}$. This helps ensure proper scaling for the price approximation with no need to rescale the outputs by hand and is motivated by the following observation:

$$\mathbb{E}[e^{-rT} Z^\Xi | \Xi, K, T] = X_0 \underbrace{\mathbb{E}\left[e^{-rT} \frac{X_T^\Xi}{X_0} \mathbb{1}_{\{X_T^\Xi \ge K\}} | \Xi, K, T \right]}_{\in [0,1]} - e^{-rT} K\, \mathbb{Q}(X_T^\Xi \ge K \,|\, \Xi, K, T)$$

5.14. From a GPU programming perspective, a size of 32 is attractive, especially for Nvidia GPUs since a warp is of size 32 [NVIDIA Corporation, 2020b], as it gives the possibility to have threads inside the same warp work on the same model and product parameters and thus prevent thread divergences that would otherwise occur if for example two threads of the same warp worked on two different realizations of the maturity T. It also allows to compute the empirical average over the 32 conditional realizations using only warp intrinsics and without resorting to shared memory or synchronization barriers.

5.5.2 Execution Times and Benchmarks

The experiments are done on a single Nvidia V100 GPU, although one could readily extend our implementation to multi-GPU training using lock-free approaches such as Hogwild training [Recht et al., 2011]. The simulations are fast as no pricing is being performed and path-wise sensitivities of the payoffs are computed with the forward approach described in Example 5.8. Both are implemented on GPU in `CUDA` and take ∼1 min 20 secs.

To demonstrate the effectiveness of our approach in terms of speed and memory footprint, we reported in tables 5.2 and 5.3 the execution time and memory usage when performing one iteration of back-propagation, *i.e.* the library's AAD with respect to network weights, through a neural network with 5 layers and respectively 64 and 128 neurons per layer, using a loss function involving either a sum of errors of each derivative of the network with respect to its inputs (*i.e.* similarly to the second sum in (5.4)) or one error involving a random directional derivative (*i.e.* as in the LHS of (5.8)). The last two rows correspond to our proposed complex-step differentiation approach and refer to respectively using `pytorch`'s Wirtinger-based AAD for complex functions and our custom `C++` implementation specializing in holomorphic functions. For the sake of accuracy, the other rows are all implemented in `C++` but very similar timings were also achieved using simply `pytorch`'s *just-in-time* compilation mechanism in plain Python.

Training for 300 epochs, with 512 batch iterations per epoch (amounting in total to 153600 SGD iterations), takes ∼13mins (compared to at least hours for equivalent accuracy when training using full gradients). Evaluating 1024 prices along with their 28 derivatives during inference takes ∼5ms on an Nvidia T4 GPU, which is very appealing for model calibration routines.

	Time	**Cumul. speedup**	**Mem. usage**
Full gradients, backward	17.01(0.40)		2365.68
Full gradients, forward	16.56(0.48)	×1.03	1330.74
Exact directional derivatives	6.42(0.14)	×2.65	433.07
CSD directional derivatives	2.60(0.19)	×6.54	139.74
CSD directional derivatives, `C++`	**1.75(0.27)**	**×9.72**	**90.20**

Table 5.2. Time spent (in ms), cumulative speed-ups and memory usage (in MB) during 1 iteration of back-propagation for different differentiation procedures for a neural network with 64 neurons/layer and 5 hidden layers, assuming 28-dimensional input and scalar outputs.

	Time	**Cumul. speedup**	**Mem. usage**
Full gradients, backward	59.65(1.00)		8834.49
Full gradients, forward	48.42(0.48)	×1.23	3169.55
Exact directional derivatives	18.86(0.16)	×3.16	1343.92
CSD directional derivatives	5.17(0.04)	×11.54	276.92
CSD directional derivatives, `C++`	**2.49(0.28)**	**×23.95**	**178.00**

Table 5.3. Time spent (in ms), cumulative speed-ups and memory usage (in MB) during 1 iteration of back-propagation for different differentiation procedures for a neural network with 128 neurons/layer and 5 hidden layers, assuming 28-dimensional input and scalar outputs.

5.5.3 Validation Without Ground-truth Values

Even though we don't compute *ground-truth* prices, *e.g.* using a dedicated Monte-Carlo simulation for each set of model and product parameters, as we are only simulating payoffs (or a conditional sample average over a very small sample), one can still estimate an L^2 distance to the ground-truth prices by following a *twin-simulation* procedure introduced in [Abbas-Turki et al., 2021]. The idea is to notice that if φ is our neural network, then:

$$\begin{aligned}\mathbb{E}[(\varphi(\Xi,K,T)-\mathbb{E}[e^{-rT}Z^{\Xi}|\Xi,K,T])^2] \;&=\; \mathbb{E}[\varphi(\Xi,K,T)\,(\varphi(\Xi,K,T)-e^{-rT}\,(Z^{\Xi,1}+Z^{\Xi,2}))]\\ &\quad+\mathbb{E}[e^{-2rT}Z^{\Xi,1}Z^{\Xi,2}]\end{aligned}$$

where $Z^{\Xi,1}$ and $Z^{\Xi,2}$ are two conditionally independent copies of Z^{Ξ} given (Ξ,K,T). Hence, via two sub-simulations conditional on each realization of the model and product parameters, one can estimate the L^2 distance between the neural network approximation and the ground-truth price without ever computing the latter. Assuming a sample of size N independent of the trajectories which were used for training, we can estimate this distance using the following unbiased estimator:

$$\begin{aligned}\mathrm{MSE}_N^{\mathrm{twin}} \;:=\; \frac{1}{N}\sum_{i=1}^{N}\varphi(\Xi^{(i)},K^{(i)},T^{(i)})\left(\varphi(\Xi^{(i)},K^{(i)},T^{(i)})-e^{-r^{(i)}T^{(i)}}\left(Z^{\Xi^{(i)},1}+Z^{\Xi^{(i)},2}\right)\right)\\ +\,e^{-2r^{(i)}T^{(i)}}\,Z^{\Xi^{(i)},1}Z^{\Xi^{(i)},2}\end{aligned}$$

where $((\Xi^{(i)},K^{(i)},T^{(i)}))_{1\le i\le N}$ is an i.i.d sample of (Ξ,K,T) and for each $i\in\{1,\dots,N\}$, $Z^{\Xi^{(i)},1}$ and $Z^{\Xi^{(i)},2}$ are two conditionally independent copies of Z^{Ξ} given $(\Xi,K,T)=(\Xi^{(i)},K^{(i)},T^{(i)})$ and the sample $\left(\left(\Xi^{(i)},K^{(i)},T^{(i)},Z^{\Xi^{(i)},1},Z^{\Xi^{(i)},2}\right)\right)_{1\le i\le N}$ is independent of the sample that was used for the training.

In Table 5.4, we list the estimates we obtain for our method along with alternative approaches using only payoffs, *i.e.* approaches not seeking to project path-wise sensitivities.

	$\sqrt{\mathrm{MSE}_N^{\mathrm{twin}}}$	stdev of $\mathrm{MSE}_N^{\mathrm{twin}}$
Projecting both payoffs and path-wise sensitivities	$\mathbf{4.9\cdot 10^{-4}}$	$1.7\cdot 10^{-6}$
Projecting only payoffs, same # of samples	$7.06\cdot 10^{-3}$	$1.7\cdot 10^{-6}$
Projecting only payoffs, $2\times$ as many samples	$4.30\cdot 10^{-3}$	$1.7\cdot 10^{-6}$
Projecting only payoffs, $4\times$ as many samples	$1.42\cdot 10^{-3}$	$1.7\cdot 10^{-6}$

Table 5.4. $\mathrm{MSE}_N^{\mathrm{twin}}$ estimates for different approaches for projecting payoffs.

5.5.4 Calibration Example

We consider the problem of fitting our fixed-grid local volatility nodes by using the price approximation learned by our neural network in the calibration phase. More precisely, we seek local volatility nodes $(\sigma_{i,j})_{\substack{1\le i\le m\\1\le j\le M}}$ such that:

$$(\sigma_{i,j})_{i,j}\in\underset{(\sigma_{i,j})_{i,j}\in[0.1,2]^{[\![1,m]\!]\times[\![1,M]\!]}}{\operatorname{argmin}}\sum_{l=1}^{L}\left(\varphi((r,\sigma_{0,0},\dots,\sigma_{i,j},\dots\sigma_{m,M}),k^{(l)},\tau^{(l)})-p_{\mathrm{mkt}}^{(l)}\right)^2$$

where we assume that we have access to L market prices of vanilla calls $p_{\mathrm{mkt}}^{(1)},\dots,p_{\mathrm{mkt}}^{(L)}$ corresponding to strikes $k^{(1)},\dots,k^{(L)}$ and maturities $\tau^{(1)},\dots,\tau^{(L)}$, and for simplicity of the experiment a flat term structure of risk-free rates.

Using the same grid of size 5×5 for the local volatility as previously, we solve the above calibration problem using Tesla's stock (Nasdaq:`TSLA`) as our underlying on 2022/02/14 using a vanilla gradient descent with respect to the local volatility nodes. We show in Figure 5.4 the smiles resulting from the model parameters obtained through the neural network pricing proxy. Under given model parameters, the smiles are constructed using a slow Monte-Carlo pricing to ensure a validation that is independent of the neural network approximation. We obtain close to perfect fits when compared to the implied volatilities of the market prices.

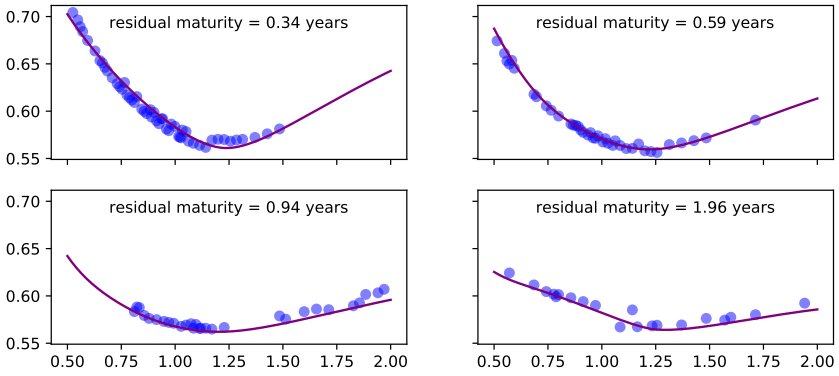

Figure 5.4. Fit of TSLA implied volatility smiles on 2022/02/14 using a 5×5 local volatility model calibrated using our proxy pricer. **Blue dots:** market prices, **purple curve:** implied volatility smile of fitted local volatility model, ***x*-axis:** moneyness, ***y*-axis:** implied volatility levels.

Although this is not the goal of the present article, notice that the flexibility to specify for instance custom grids for the local volatility, and easily by extension stochastic-local volatility models [Guyon and Henry-Labordere, 2011; Tian et al., 2015], gives a new possibility to regularize calibration problems by enforcing parsimony directly at the grid level.

The calibration approach is entirely orthogonal to the procedure presented in this paper. One can imagine more elaborate calibration procedures based on Levenberg-Marquardt or other algorithms. The main idea remains that inside the calibration procedure, the pricing function is replaced by the approximation given by our neural network, and, whenever derivatives are needed, these can be computed either explicitly or using automatic differentiation.

5.A Derivatives with respect to time to maturity in discrete time models

Let $\xi \in \mathbb{R}^n$ and $r, \tau, k > 0$. In our general SDE (5.1) we will assume one single spatial dimension in X_t (*i.e.* $d = 1$) and we will consider the payoff of a vanilla call with time to maturity τ and assume risk-free rate r. Let $h(x) = (x-k)^+$. For every $t > 0$, let $x \mapsto p(t,x)$ be the probability density of X_t^ξ. Then, for all $x \in \mathbb{R}$, the density p follows the following Fokker-Planck equation at (τ, x):

$$\partial_\tau p(\tau,x) = -\partial_x(f(\tau,x,\xi)\, p(\tau,x)) + \frac{1}{2}\partial_{xx}(g(\tau,x,\xi)^2\, p(\tau,x)) \tag{5.10}$$

We will assume in the following that X_τ^ξ is at least L^3-integrable. In this case the Lipschitz regularity of f and g in their second argument yields:

$$\begin{aligned}\lim_{x\to+\infty} x\, f(\tau,x,\xi)\, p(\tau,x) &= 0\\ \lim_{x\to+\infty} g(\tau,x,\xi)^2\, p(\tau,x) &= 0\\ \lim_{x\to+\infty} x\, \partial_x(g(\tau,x,\xi)^2\, p(\tau,x)) &= 0\end{aligned}$$

Then using integration by parts, we have:

$$\begin{aligned}\int_{-\infty}^{+\infty} h(x)\, \partial_x(f(\tau,x,\xi)\, p(\tau,x))\,\mathrm{d}x &= e^{-r\tau}\int_{-\infty}^{+\infty}(x-k)\, \partial_x(f(\tau,x,\xi)\, p(\tau,x))\,\mathrm{d}x\\ &= -e^{-r\tau}\mathbb{E}\big[\, f(\tau,X_\tau^\xi,\xi)\, \mathbb{1}_{\{X_\tau^\xi\geq k\}}\big] \qquad (5.11)\end{aligned}$$

and

$$\begin{aligned}\int_{-\infty}^{+\infty} h(x)\, \partial_{xx}(g(\tau,x,\xi)^2\, p(\tau,x))\,\mathrm{d}x &= e^{-r\tau}\int_{-\infty}^{+\infty}(x-k)\, \partial_{xx}(g(\tau,x,\xi)^2\, p(\tau,x))\,\mathrm{d}x\\ &= e^{-r\tau} g(\tau,k,\xi)^2\, p(\tau,k)\end{aligned}$$

We also have via the Breeden-Litzenberger formula that $p(\tau,k) = \partial_{kk}\mathbb{E}[(X_\tau^\xi - k)^+]$, thus:

$$\int_{-\infty}^{+\infty} h(x)\, \partial_{xx}(g(\tau,x,\xi)^2\, p(\tau,x))\,\mathrm{d}x = e^{-r\tau} g(\tau,k,\xi)^2\, \partial_{kk}\mathbb{E}[(X_\tau^\xi - k)^+] \qquad (5.12)$$

Notice now that we have:

$$\partial_\tau(e^{-r\tau}\mathbb{E}[(X_\tau^\xi - k)^+]) = -r\, e^{-r\tau}\mathbb{E}[(X_\tau^\xi - k)^+] + e^{-r\tau}\int_k^{+\infty}(x-k)\, \partial_\tau p(\tau,x)\,\mathrm{d}x \qquad (5.13)$$

Plugging the expressions of (5.11), (5.12) and (5.13) into the Fokker-Planck equation in (5.10), we get:

$$\begin{aligned}\partial_\tau(e^{-r\tau}\mathbb{E}[(X_\tau^\xi - k)^+]) &= -r\, e^{-r\tau}\mathbb{E}[(X_\tau^\xi - k)^+] + e^{-r\tau}\mathbb{E}\big[\, f(\tau,X_\tau^\xi,\xi)\, \mathbb{1}_{\{X_\tau^\xi\geq k\}}\big]\\ &\quad + \frac{1}{2}\, e^{-r\tau} g(\tau,k,\xi)^2\, \partial_{kk}\mathbb{E}[(X_\tau^\xi - k)^+] \qquad (5.14)\end{aligned}$$

Finally notice that for any $0 < h < \tau$, applying the tower property and the Breeden-Litzenberger formula conditionally at $\tau - h$ we have:

$$\begin{aligned}\partial_{kk}\mathbb{E}[(X_\tau^\xi - k)^+] &= \mathbb{E}[\partial_{kk}\mathbb{E}[(X_\tau^\xi - k)^+ | X_{\tau-h}^\xi]]\\ &= \mathbb{E}[p^h(\tau,k|X_{\tau-h}^\xi)]\end{aligned}$$

where $x \mapsto p^h(\tau,x|x')$ is the density of X_τ^ξ conditional on $X_{\tau-h}^\xi = x'$.

Thus, going back to random model parameters, thanks to (5.14) one can write the derivative of the price with respect to time to maturity as a conditional expectation involving either only one second spatial derivative, or no derivatives at all if one knows the conditional density $p^h(\tau,.|.)$ of $X_\tau^\xi | X_{\tau-h}^\xi$. The latter can in practice be replaced, given sufficiently small h, with the closed-form available in discrete time given that increments in an Euler scheme for example are Gaussian (or log-normal if one discretizes the log-dynamics). We refer to works such as [Bally and Talay, 1996] for a quantification of the error when using the density of the discrete solution instead of that of the continuous time solution.

We used this approach in our implementation for the derivative with respect to time to maturity, by considering this integrand:

$$r\,e^{-r\tau}\left(-(X_\tau^\xi-k)^+ + f(\tau,X_\tau^\xi,\xi)\,\mathbb{1}_{\{X_\tau^\xi\geq k\}} + \frac{1}{2}\,g(\tau,k,\xi)^2\,p^h(\tau,k|X_{\tau-h}^\xi)\right)$$

as a *fictitious* path-wise sensitivity with respect to time to maturity.

5.B Differentiation of the local volatility function in Example 5.8

We resume here the calculations of Example 5.8. We have:

- If there exist $p\in\{1,\dots,m-1\}$ and $q\in\{1,\dots,M-1\}$ such that $t^{(p)}\leq t_i<t^{(p+1)}$ and $s^{(q)}\leq\hat{s}_i<s^{(q+1)}$, then:

$$\partial_{u,v}\sigma(t_i,\hat{s}_i)=\begin{cases}\dfrac{(t^{(p+1)}-t)(s^{(q+1)}-\hat{s}_i)}{(t^{(p+1)}-t^{(p)})(s^{(q+1)}-s^{(q)})} & \text{if}\quad (u,v)=(p,q)\\ \dfrac{(t-t^{(p)})\,(s^{(q+1)}-\hat{s}_i)}{(t^{(p+1)}-t^{(p)})\,(s^{(q+1)}-s^{(q)})} & \text{if}\quad (u,v)=(p+1,q)\\ \dfrac{(t^{(p+1)}-t)\,(\hat{s}_i-s^{(q)})}{(t^{(p+1)}-t^{(p)})\,(s^{(q+1)}-s^{(q)})} & \text{if}\quad (u,v)=(p,q+1)\\ \dfrac{(t-t^{(p)})\,(\hat{s}_i-s^{(q)})}{(t^{(p+1)}-t^{(p)})\,(s^{(q+1)}-s^{(q)})} & \text{if}\quad (u,v)=(p+1,q+1)\\ 0 & \text{otherwise}\end{cases}$$

- If $t\geq t^{(m)}$ and there exists $q\in\{1,\dots,M-1\}$ such that $s^{(q)}\leq\hat{s}_i<s^{(q+1)}$, then:

$$\partial_{u,v}\sigma(t_i,\hat{s}_i)=\begin{cases}\dfrac{s^{(q+1)}-\hat{s}_i}{s^{(q+1)}-s^{(q)}} & \text{if}\quad (u,v)=(m,q)\\ \dfrac{\hat{s}_i-s^{(q+1)}}{s^{(q+1)}-s^{(q)}} & \text{if}\quad (u,v)=(m,q+1)\\ 0 & \text{otherwise}\end{cases}$$

- If $t<t^{(0)}$ and there exists $q\in\{1,\dots,M-1\}$ such that $s^{(q)}\leq\hat{s}_i<s^{(q+1)}$, then:

$$\partial_{u,v}\sigma(t_i,\hat{s}_i)=\begin{cases}\dfrac{s^{(q+1)}-\hat{s}_i}{s^{(q+1)}-s^{(q)}} & \text{if}\quad (u,v)=(0,q)\\ \dfrac{\hat{s}_i-s^{(q+1)}}{s^{(q+1)}-s^{(q)}} & \text{if}\quad (u,v)=(0,q+1)\\ 0 & \text{otherwise}\end{cases}$$

- If $s\geq s^{(M)}$ and there exists $p\in\{1,\dots,m-1\}$ such that $t^{(p)}\leq t_i<t^{(p+1)}$, then:

$$\partial_{u,v}\sigma(t_i,\hat{s}_i)=\begin{cases}\dfrac{t^{(p+1)}-t}{t^{(p+1)}-t^{(p)}} & \text{if}\quad (u,v)=(p,M)\\ \dfrac{t-t^{(p)}}{t^{(p+1)}-t^{(p)}} & \text{if}\quad (u,v)=(p+1,M)\\ 0 & \text{otherwise}\end{cases}$$

- If $s < s^{(0)}$ and there exists $p \in \{1, \dots, m-1\}$ such that $t^{(p)} \leq t_i < t^{(p+1)}$, then:

$$\partial_{u,v}\sigma(t_i, \hat{s}_i) = \begin{cases} \dfrac{t^{(p+1)} - t}{t^{(p+1)} - t^{(p)}} & \text{if} \quad (u,v) = (p,0) \\ \dfrac{t - t^{(p)}}{t^{(p+1)} - t^{(p)}} & \text{if} \quad (u,v) = (p+1,0) \\ 0 & \text{otherwise} \end{cases}$$

Bibliography

[Abadi et al., 2016] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. TensorFlow: A System for Large-Scale Machine Learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016.

[Abbas-Turki et al., 2021] Abbas-Turki, L., Crépey, S., and Saadeddine, B. Hierarchical Simulation for Learning With Defaults, 2021. Working paper available on https://www.lpsm.paris/pageperso/crepey.

[Abbas-Turki et al., 2018] Abbas-Turki, L. A., Crépey, S., and Diallo, B. XVA principles, nested Monte Carlo strategies, and GPU optimizations. *International Journal of Theoretical and Applied Finance*, 21(06):1850030, 2018.

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

[Abu-El-Haija et al., 2016] Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., and Vijayanarasimhan, S. Youtube-8m: A large-scale video classification benchmark. *arXiv preprint arXiv:1609.08675*, 2016.

[Acerbi and Tasche, 2002] Acerbi, C. and Tasche, D. On the coherence of expected shortfall. *Journal of Banking and Finance*, 26:1487–1503, 2002.

[Adomavicius and Tuzhilin, 2005] Adomavicius, G. and Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering*, 17(6):734–749, 2005.

[Agarwal et al., 2019] Agarwal, A., Marco, S. D., Gobet, E., López-Salas, J., Noubiagain, F., and Zhou, A. Numerical approximations of McKean Anticipative Backward Stochastic Differential Equations arising in Initial Margin requirements. *ESAIM: Proceedings and Surveys*, 65:1–26, 2019.

[Akiba et al., 2019] Akiba, T., Sano, S., Yanase, T., Ohta, T., and Koyama, M. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2623–2631, 2019.

[Albanese et al., 2020] Albanese, C., Armenti, Y., and Crépey, S. XVA Metrics for CCP optimisation. *Statistics & Risk Modeling*, 37(1-2):25–53, 2020.

[Albanese et al., 2017] Albanese, C., Caenazzo, S., and Crépey, S. Credit, Funding, Margin, and Capital Valuation Adjustments for Bilateral Portfolios. *Probability, Uncertainty and Quantitative Risk*, 2(7):26 pages, 2017.

[Albanese et al., 2021] Albanese, C., Crépey, S., Hoskinson, R., and Saadeddine, B. XVA Analysis From the Balance Sheet. *Quantitative Finance*, 21(1):99–123, 2021.

[Andersen et al., 2019] Andersen, L., Duffie, D., and Song, Y. Funding Value Adjustments. *Journal of Finance*, 74(1):145–192, 2019.

[Andersen et al., 2017] Andersen, L., Pykhtin, M., and Sokol, A. Rethinking the margin period of risk. *Journal of Credit Risk*, 13(1):1–45, 2017.

[Artzner et al., 2020] Artzner, P., Eisele, K.-T., and Schmidt, T. No arbitrage in insurance and the QP-rule, 2020. Working paper available as arXiv:2005.11022.

[Bally and Talay, 1996] Bally, V. and Talay, D. The law of the Euler scheme for stochastic differential equations: II. Convergence rate of the density. *Monte Carlo Methods and Applications*, 2(2):93–128, 1996.

[Barrera, 2022] Barrera, D. Confidence intervals for nonparametric regression. *arXiv preprint arXiv:2203.10643*, 2022.

[Barrera et al., 2019] Barrera, D., Crépey, S., Diallo, B., Fort, G., Gobet, E., and Stazhynski, U. Stochastic Approximation Schemes for Economic Capital and Risk Margin Computations. *ESAIM: Proceedings and Surveys*, 65:182–218, 2019.

[Barrera et al., 2022] Barrera, D., Crépey, S., Gobet, E., Nguyen, H.-D., and Saadeddine, B. Learning Value-at-Risk and Expected Shortfall, 2022. Working paper available on https://www.lpsm.paris/pageperso/crepey.

[Baydin et al., 2018] Baydin, A. G., Pearlmutter, B. A., Radul, A. A., and Siskind, J. M. Automatic differentiation in machine learning: a survey. *Journal of Marchine Learning Research*,

18:1–43, 2018.

[Bayer et al., 2016] BAYER, C., FRIZ, P., and GATHERAL, J. Pricing under rough volatility. *Quantitative Finance*, 16(6):887–904, 2016.

[Bayer and Stemper, 2018] BAYER, C. and STEMPER, B. Deep calibration of rough stochastic volatility models. *arXiv preprint arXiv:1810.03399*, 2018. https://arxiv.org/abs/1810.03399.

[Beck et al., 2019] BECK, C., BECKER, S., CHERIDITO, P., JENTZEN, A., and NEUFELD, A. Deep splitting method for parabolic PDEs, 2019. ArXiv:1907.03452.

[Becker, 2020] BECKER, L. FVA losses back in spotlight after coronavirus stress, 2020. https://www.risk.net/derivatives/7526696/fva-losses-back-in-spotlight-after-coronavirus-stress.

[Becker et al., 2019] BECKER, S., CHERIDITO, P., and JENTZEN, A. Deep optimal stopping. *Journal of Machine Learning Research*, 20:74, 2019.

[Bengio et al., 2016] BENGIO, Y., COURVILLE, A., and GOODFELLOW, I. *Deep learning*. MIT press Cambridge, 2016.

[Bergstra and Bengio, 2012] BERGSTRA, J. and BENGIO, Y. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

[Bichuch et al., 2018] BICHUCH, M., CAPPONI, A., and STURM, S. Arbitrage-Free XVA. *Mathematical Finance*, 28(2):582–620, 2018.

[Bielecki and Rutkowski, 2002] BIELECKI, T. and RUTKOWSKI, M. *Credit Risk: Modeling, Valuation and Hedging*. Springer Finance, Berlin, 2002.

[Bielecki and Rutkowski, 2015] BIELECKI, T. R. and RUTKOWSKI, M. Valuation and Hedging of Contracts with Funding Costs and Collateralization. *SIAM Journal on Financial Mathematics*, 6:594–655, 2015.

[Bondell et al., 2010] BONDELL, H., REICH, B., and WANG, H. Noncrossing quantile regression curve estimation. *Biometrika*, 97(4):825–838, 2010.

[Borthakur, 2007] BORTHAKUR, D. The hadoop distributed file system: Architecture and design. *Hadoop Project Website*, 11(2007):21, 2007.

[Bottou, 2010] BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[Bouboulis, 2010] BOUBOULIS, P. Wirtinger's calculus in general Hilbert spaces. *arXiv preprint arXiv:1005.5170*, 2010.

[Bouchard and Élie, 2008] BOUCHARD, B. and ÉLIE, R. Discrete time approximation of decoupled forward-backward SDE with jumps. *Stochastic Processes and Applications*, 118(1):53–75, 2008.

[Bouchard and Touzi, 2004] BOUCHARD, B. and TOUZI, N. Discrete-time approximation and Monte-Carlo simulation of backward stochastic differential equations. *Stochastic Processes and their applications*, 111(2):175–206, 2004.

[Bozinovski, 2020] BOZINOVSKI, S. Reminder of the first paper on transfer learning in neural networks, 1976. *Informatica*, 44(3), 2020.

[Bradbury et al., 2018] BRADBURY, J., FROSTIG, R., HAWKINS, P., JOHNSON, M. J., LEARY, C., MACLAURIN, D., NECULA, G., PASZKE, A., VANDERPLAS, J., WANDERMAN-MILNE, S., and ZHANG, Q. JAX: composable transformations of Python+NumPy programs, 2018. http://github.com/google/jax.

[Brigo and Capponi, 2010] BRIGO, D. and CAPPONI, A. Bilateral counterparty risk with application to CDSs. *Risk Magazine*, pages March 85–90, 2010. Preprint version available at https://arxiv.org/abs/0812.3705.

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

[Broadie and Glasserman, 1996] BROADIE, M. and GLASSERMAN, P. Estimating security price derivatives using simulation. *Management science*, 42(2):269–285, 1996.

[Buehler et al., 2019] BUEHLER, H., GONON, L., TEICHMANN, J., and WOOD, B. Deep hedging. *Quantitative Finance*, 19(8):1271–1291, 2019.

[Burgard and Kjaer, 2011] BURGARD, C. and KJAER, M. In the balance. *Risk Magazine*, pages October 72–75, 2011.

[Burgard and Kjaer, 2013] BURGARD, C. and KJAER, M. Funding Costs, Funding Strategies. *Risk Magazine*, pages December 82–87, 2013. Preprint version available at https://ssrn.com/abstract=2027195.

[Burgard and Kjaer, 2017] BURGARD, C. and KJAER, M. Derivatives funding, netting and accounting. *Risk Magazine*, pages March 100–104, 2017. Preprint version available at https://ssrn.com/abstract=2534011.

[Cannon, 2018] Cannon, A. J. Non-crossing nonlinear regression quantiles by monotone composite quantile regression neural network, with application to rainfall extremes. *Stochastic environmental research and risk assessment*, 32(11):3207–3225, 2018.

[Carmona and Crépey, 2010] Carmona, R. and Crépey, S. Particle methods for the estimation of credit portfolio loss distributions. *International Journal of Theoretical and Applied Finance*, 13(04):577–602, 2010.

[Castagna, 2014] Castagna, A. Towards a theory of internal valuation and transfer pricing of products in a bank: Funding, credit risk and economic capital, 2014. Available at http://ssrn.com/abstract=2392772.

[Cesari et al., 2010] Cesari, J., Aquilina, J., and Charpillon, N. *Modelling, Pricing, and Hedging Counterparty Credit Exposure*. Springer, 2010. ISBN 9783642044847. https://books.google.com/books?id=kb38tDwznN4C.

[Chalapathy and Chawla, 2019] Chalapathy, R. and Chawla, S. Deep learning for anomaly detection: A survey. *arXiv preprint arXiv:1901.03407*, 2019.

[Chan et al., 1999] Chan, P. K., Fan, W., Prodromidis, A. L., and Stolfo, S. J. Distributed data mining in credit card fraud detection. *IEEE Intelligent Systems and Their Applications*, 14(6):67–74, 1999.

[Chassagneux and Crisan, 2014] Chassagneux, J.-F. and Crisan, D. Runge–Kutta schemes for backward stochastic differential equations. *The Annals of Applied Probability*, 24(2):679–720, 2014.

[Chassagneux et al., 2019] Chassagneux, J.-F., Crisan, D., and Delarue, F. Numerical method for FBSDEs of McKean-Vlasov type. *The Annals of Applied Probability*, 29(3):1640–1684, 2019.

[Chassagneux and Richou, 2016] Chassagneux, J.-F. and Richou, A. Numerical simulation of quadratic BSDEs. *The Annals of Applied Probability*, 26(1):262–304, 2016.

[Chassagneux and Richou, 2019] Chassagneux, J.-F. and Richou, A. Rate of convergence for the discrete-time approximation of reflected BSDEs arising in switching problems. *Stochastic Processes and their Applications*, 129(11):4597–4637, 2019.

[Chen et al., 2016] Chen, J., Pan, X., Monga, R., Bengio, S., and Jozefowicz, R. Revisiting distributed synchronous SGD. *arXiv preprint arXiv:1604.00981*, 2016.

[Chen, 2007] Chen, X. Large sample sieve estimation of semi-nonparametric models. *Handbook of econometrics*, 6:5549–5632, 2007.

[Cohen et al., 2016] Cohen, N., Sharir, O., and Shashua, A. On the expressive power of deep learning: A tensor analysis. In *Conference on learning theory*, pages 698–728. PMLR, 2016.

[Collin-Dufresne et al., 2004] Collin-Dufresne, P., Goldstein, R., and Hugonnier, J. A general formula for valuing defaultable securities. *Econometrica*, 72(5):1377–1407, 2004.

[Committee of European Insurance and Occupational Pensions Supervisors, 2010] Committee of European Insurance and Occupational Pensions Supervisors. QIS5 Technical Specifications, 2010. https://eiopa.europa.eu/Publications/QIS/QIS5-technical_specifications_20100706.pdf.

[Crépey, 2013] Crépey, S. *Financial Modeling: A Backward Stochastic Differential Equations Perspective*. Springer Finance. Springer, 2013. ISBN 9783642371127.

[Crépey, 2015] Crépey, S. Bilateral counterparty risk under funding constraints. Part I: Pricing, followed by Part II: CVA. *Mathematical Finance*, 25(1):1–22 and 23–50, 2015. First published online on 12 December 2012.

[Crépey, 2022] Crépey, S. The Cost-of-Capital XVA Approach in Continuous Time—Part I: Positive XVAs, and Part II: Cash Flows Arithmetics, 2022. Working papers available at https://perso.lpsm.paris/∼crepey.

[Crépey et al., 2014] Crépey, S., Bielecki, T. R., and Brigo, D. *Counterparty Risk and Funding: A Tale of Two Puzzles*. Taylor & Francis, New York, 2014. Chapman & Hall/CRC Financial Mathematics Series.

[Crépey et al., 2020] Crépey, S., Sabbagh, W., and Song, S. When Capital Is a Funding Source: The Anticipated Backward Stochastic Differential Equations of X-Value Adjustments. *SIAM Journal on Financial Mathematics*, 11(1):99–130, 2020.

[Crépey and Song, 2015] Crépey, S. and Song, S. BSDEs of Counterparty Risk. *Stochastic Processes and their Applications*, 125(8):3023–3052, 2015.

[Crépey and Song, 2016] Crépey, S. and Song, S. Counterparty Risk and Funding: Immersion and Beyond. *Finance and Stochastics*, 20(4):901–930, 2016.

[Crépey and Song, 2017] Crépey, S. and Song, S. Invariance Times. *The Annals of Probability*, 45(6B):4632–4674, 2017.

[Crisan et al., 2010] Crisan, D., Manolarakis, K., and Touzi, N. On the Monte Carlo simulation of BSDEs: An improvement on the Malliavin weights. *Stochastic Processes and their Applications*,

120(7):1133–1158, 2010.

[Cybenko, 1989] CYBENKO, G. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, 2(4):303–314, 1989.

[Czarnecki et al., 2017] CZARNECKI, W. M., OSINDERO, S., JADERBERG, M., SWIRSZCZ, G., and PASCANU, R. Sobolev training for neural networks. *Advances in Neural Information Processing Systems*, 30, 2017.

[De Spiegeleer et al., 2018] DE SPIEGELEER, J., MADAN, D. B., REYNERS, S., and SCHOUTENS, W. Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 18(10):1635–1643, 2018.

[Delarue and Menozzi, 2006] DELARUE, F. and MENOZZI, S. A forward–backward stochastic algorithm for quasi-linear PDEs. *The Annals of Applied Probability*, 16(1):140–184, 2006.

[Deng et al., 2009] DENG, J., DONG, W., SOCHER, R., LI, L.-J., LI, K., and FEI-FEI, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[Dimitriadis and Bayer, 2019] DIMITRIADIS, T. and BAYER, S. A joint quantile and expected shortfall regression framework. *Electronic Journal of Statistics*, 13(1):1823–1871, 2019.

[Duffie and Huang, 1996] DUFFIE, D. and HUANG, M. Swap Rates and Credit Quality. *The Journal of Finance*, 51(3):921–949, 1996.

[Duffie and Sharer, 1986] DUFFIE, D. and SHARER, W. Equilibrium and the Role of the Firm in Incomplete Market, 1986. Stanford University, Working Paper No. 915, available at https://www.gsb.stanford.edu/faculty-research/working-papers/equilibrium-role-firm-incomplete-markets.

[Dybvig, 1992] DYBVIG, P. Hedging non-traded wealth: when is there separation of hedging and investment. In HODGES, S. (editor), *Options: recent advances in theory and practice*, volume 2, pages 13–24. Manchester University Press, 1992.

[E et al., 2017] E, W., HAN, J., and JENTZEN, A. Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations. *Communications in Mathematics and Statistics*, 5(4):370–398, 2017.

[Ekvall, 1996] EKVALL, N. A lattice approach for pricing of multivariate contingent claims. *European Journal of Operational Research*, 91(2):214–228, 1996.

[El Karoui et al., 1997] EL KAROUI, N., PENG, S., and QUENEZ, M.-C. Backward stochastic differential equations in finance. *Mathematical Finance*, 7:1–71, 1997.

[Eldan and Shamir, 2016] ELDAN, R. and SHAMIR, O. The power of depth for feedforward neural networks. In *Conference on learning theory*, pages 907–940. PMLR, 2016.

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

[Elouerkhaoui, 2017] ELOUERKHAOUI, Y. *Credit Correlation: Theory and Practice*. Palgrave Macmillan, 2017.

[Fissler et al., 2016] FISSLER, T., ZIEGEL, J., and GNEITING, T. Expected Shortfall is jointly elicitable with Value at Risk—Implications for backtesting. *Risk Magazine*, page January, 2016.

[Fissler and Ziegel, 2016] FISSLER, T. and ZIEGEL, J. F. Higher order elicitability and Osband's principle. *The Annals of Statistics*, 44(4):1680–1707, 2016.

[Föllmer and Schied, 2016] FÖLLMER, H. and SCHIED, A. *Stochastic Finance: An Introduction in Discrete Time*. De Gruyter Graduate, Berlin, 4th edition, 2016.

[Gasthaus et al., 2019] GASTHAUS, J., BENIDIS, K., WANG, Y., RANGAPURAM, S. S., SALINAS, D., FLUNKERT, V., and JANUSCHOWSKI, T. Probabilistic forecasting with spline quantile function RNNs. In *The 22nd international conference on artificial intelligence and statistics*, pages 1901–1910. PMLR, 2019.

[Glasserman, 2004] GLASSERMAN, P. *Monte Carlo methods in financial engineering*, volume 53. Springer, 2004.

[Gnoatto et al., 2021] GNOATTO, A., REISINGER, C., and PICARELLI, A. Deep xVA solver–a neural network based counterparty credit risk management framework. *arXiv:2005.02633*, 2021.

[Gobet, 2016] GOBET, E. *Monte-Carlo methods and stochastic processes: from linear to non-linear*. Chapman and Hall/CRC, 2016.

[Gobet et al., 2005] GOBET, E., LEMOR, J.-P., and WARIN, X. A Regression-based Monte Carlo method to solve Backward Stochastic Differential Equations. *The Annals of Applied Probability*, 15(3):2172–2202, 2005.

[Goodfellow et al., 2016] GOODFELLOW, I., BENGIO, Y., and COURVILLE, A. *Deep Learning*. MIT Press, 2016.

[Gottardi, 1995] GOTTARDI, P. An analysis of the conditions for the validity of Modigliani-Miller Theorem with incomplete markets. *Economic Theory*, 5:191–207, 1995.

[Goudenege et al., 2020] Goudenege, L., Molent, A., and Zanette, A. Machine learning for pricing American options in high-dimensional Markovian and non-Markovian models. *Quantitative Finance*, 20(4):573–591, 2020.

[Green et al., 2014] Green, A., Kenyon, C., and Dennis, C. KVA: capital valuation adjustment by replication. *Risk Magazine*, pages December 82–87, 2014.

[Guyon and Henry-Labordere, 2011] Guyon, J. and Henry-Labordere, P. The smile calibration problem solved. *Available at SSRN 1885032*, 2011.

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

[Hatalis et al., 2017] Hatalis, K., Lamadrid, A. J., Scheinberg, K., and Kishore, S. Smooth pinball neural network for probabilistic forecasting of wind power. *arXiv preprint arXiv:1710.01720*, 2017.

[He et al., 2015] He, K., Zhang, X., Ren, S., and Sun, J. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[He, 1997] He, X. Quantile curves without crossing. *The American Statistician*, 51(2):186–192, 1997.

[Henry-Labordère, 2017] Henry-Labordère, P. Deep primal-dual algorithm for BSDEs: applications of machine learning to CVA and IM, 2017. Available at SSRN: http://ssrn.com/abstract=3071506.

[Henry-Labordere et al., 2017] Henry-Labordere, P., Tan, X., and Touzi, N. Unbiased simulation of stochastic differential equations. *The Annals of Applied Probability*, 27(6):3305–3341, 2017.

[Hernandez, 2016] Hernandez, A. Model calibration with neural networks. *Available at SSRN 2812140*, 2016.

[Hoeting et al., 1999] Hoeting, J. A., Madigan, D., Raftery, A. E., and Volinsky, C. T. Bayesian Model Averaging: A Tutorial. *Statistical Science*, 14(4):382–417, 1999.

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

[Horvath et al., 2021] Horvath, B., Muguruza, A., and Tomas, M. Deep learning volatility: a deep neural network perspective on pricing and calibration in (rough) volatility models. *Quantitative Finance*, 21(1):11–27, 2021.

[Huge and Savine, 2020] Huge, B. N. and Savine, A. Differential machine learning. *Available at SSRN 3591734*, 2020.

[Hull and White, 2012] Hull, J. and White, A. The FVA debate, followed by The FVA debate continued. *Risk Magazine*, pages July 83–85 and October 52, 2012.

[Huré et al., 2020] Huré, C., Pham, H., and Warin, C. Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation*, 89(324):1547–1579, 2020.

[Huré et al., 2020] Huré, C., Pham, H., and Warin, X. Deep backward schemes for high-dimensional nonlinear PDEs. *Mathematics of Computation*, 89(324):1547–1579, 2020.

[International Financial Reporting Standards, 2013] International Financial Reporting Standards. IFRS 4 insurance contracts exposure draft, 2013.

[Jacod, 1979] Jacod, J. *Calcul Stochastique et Problèmes de Martingales*. Lecture Notes Math. 714. Springer, 1979.

[Janai et al., 2020] Janai, J., Güney, F., Behl, A., Geiger, A., et al. Computer vision for autonomous vehicles: Problems, datasets and state of the art. *Foundations and Trends® in Computer Graphics and Vision*, 12(1–3):1–308, 2020.

[Kallenberg, 2006] Kallenberg, O. *Foundations of modern probability*. Springer, 2006.

[Kidger and Lyons, 2020] Kidger, P. and Lyons, T. Universal approximation with deep narrow networks. In *Conference on learning theory*, pages 2306–2327. PMLR, 2020.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Kjaer, 2019] Kjaer, M. In the balance redux. *Risk Magazine*, (November), 2019.

[Koenker, 2004] Koenker, R. Quantile regression for longitudinal data. *Journal of Multivariate Analysis*, 91(1):74–89, 2004.

[Koenker, 2017] Koenker, R. Quantile regression: 40 years on. *Annual Review of Economics*, 9:155–176, 2017.

[Koenker and Park, 1996] Koenker, R. and Park, B. J. An interior point algorithm for nonlinear quantile regression. *Journal of Econometrics*, 71(1-2):265–283, 1996.

[Kuznetsova et al., 2020] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Kolesnikov, A., et al. The open images dataset v4. *International Journal of Computer Vision*, 128(7):1956–1981, 2020.

[Lakshman and Malik, 2010] LAKSHMAN, A. and MALIK, P. Cassandra: a decentralized structured storage system. *ACM SIGOPS Operating Systems Review*, 44(2):35–40, 2010.

[Lang, 2003] LANG, S. *Complex analysis*, volume 103. Springer Science & Business Media, 2003.

[LeCun et al., 2015] LECUN, Y., BENGIO, Y., and HINTON, G. Deep learning. *nature*, 521(7553):436–444, 2015.

[Lessmann et al., 2015] LESSMANN, S., BAESENS, B., SEOW, H.-V., and THOMAS, L. C. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136, 2015.

[Li et al., 2017] LI, L., JAMIESON, K., DESALVO, G., ROSTAMIZADEH, A., and TALWALKAR, A. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*, 18(1):6765–6816, 2017.

[Liu and Wu, 2011] LIU, Y. and WU, Y. Simultaneous multiple non-crossing quantile regression estimation using kernel constraints. *Journal of nonparametric statistics*, 23(2):415–437, 2011.

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

[Luebke, 2008] LUEBKE, D. CUDA: Scalable parallel programming for high-performance scientific computing. In *2008 5th IEEE international symposium on biomedical imaging: from nano to macro*, pages 836–838. IEEE, 2008.

[Lyons and Victoir, 2004] LYONS, T. and VICTOIR, N. Cubature on Wiener space. *Proceedings of the Royal Society of London. Series A: Mathematical, Physical and Engineering Sciences*, 460(2041):169–198, 2004.

[Magoulas and Lorica, 2009] MAGOULAS, R. and LORICA, B. Big data: Technologies and techniques for large scale data. *Jimmy Guterman, Release*, 2:125, 2009.

[Martins et al., 2003] MARTINS, J. R., STURDZA, P., and ALONSO, J. J. The complex-step derivative approximation. *ACM Transactions on Mathematical Software (TOMS)*, 29(3):245–262, 2003.

[McCrickerd and Pakkanen, 2018] MCCRICKERD, R. and PAKKANEN, M. S. Turbocharging Monte Carlo pricing for the rough Bergomi model. *Quantitative Finance*, 18(11):1877–1886, 2018.

[Meinshausen and Ridgeway, 2006] MEINSHAUSEN, N. and RIDGEWAY, G. Quantile regression forests. *Journal of Machine Learning Research*, 7(6), 2006.

[Merton, 1974] MERTON, R. On the pricing of corporate debt: the risk structure of interest rates. *The Journal of Finance*, 29:449–470, 1974.

[Modigliani and Miller, 1958] MODIGLIANI, F. and MILLER, M. The cost of capital, corporation finance and the theory of investment. *Economic Review*, 48:261–297, 1958.

[Mohri et al., 2018] MOHRI, M., ROSTAMIZADEH, A., and TALWALKAR, A. *Foundations of machine learning*. MIT press, 2018.

[Moon et al., 2021] MOON, S. J., JEON, J.-J., LEE, J. S. H., and KIM, Y. Learning Multiple Quantiles with Neural Networks. *Journal of Computational and Graphical Statistics*, pages 1–11, 2021.

[Myers, 1977] MYERS, S. Determinants of corporate borrowing. *Journal of Financial Economics*, 5:147–175, 1977.

[Nickolls and Dally, 2010] NICKOLLS, J. and DALLY, W. J. The GPU computing era. *IEEE micro*, 30(2):56–69, 2010.

[NVIDIA Corporation, 2020a] NVIDIA CORPORATION. NVIDIA A100 Datasheet, 2020a. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/a100/pdf/nvidia-a100-datasheet.pdf.

[NVIDIA Corporation, 2020b] NVIDIA CORPORATION. Programming Guide: CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html, 2020b. Accessed: 2020-04-28.

[Omar et al., 2013] OMAR, S., NGADI, A., and JEBUR, H. H. Machine learning techniques for anomaly detection: an overview. *International Journal of Computer Applications*, 79(2), 2013.

[Padilla et al., 2020] PADILLA, O. H. M., TANSEY, W., and CHEN, Y. Quantile regression with ReLU Networks: Estimators and minimax rates. *arXiv preprint arXiv:2010.08236*, 2020.

[Pan and Yang, 2009] PAN, S. J. and YANG, Q. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[Pardoux and Peng, 1990] PARDOUX, E. and PENG, S. Adapted solution of a backward stochastic differential equation. *Systems & Control Letters*, 14(1):55–61, 1990.

[Paszke et al., 2019] PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L., et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[Peng and Yang, 2009] Peng, S. and Yang, Z. Anticipated backward stochastic differential equations. *The Annals of Probability*, 37(3):877–902, 2009.

[Piterbarg, 2010] Piterbarg, V. Funding beyond discounting: collateral agreements and derivatives pricing. *Risk Magazine*, pages August 57–63, 2010.

[Raissi, 2018] Raissi, M. Forward-backward stochastic neural networks: Deep learning of high-dimensional partial differential equations. *arXiv preprint arXiv:1804.07010*, 2018.

[Rasmussen and Williams, 2006] Rasmussen, C. E. and Williams, C. K. I. *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[Recht et al., 2011] Recht, B., Re, C., Wright, S., and Niu, F. Hogwild!: A lock-free approach to parallelizing stochastic gradient descent. *Advances in neural information processing systems*, 24, 2011.

[Robbins and Monro, 1951] Robbins, H. and Monro, S. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[Rockafellar and Uryasev, 2000] Rockafellar, R. and Uryasev, S. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[Rodrigues and Pereira, 2020] Rodrigues, F. and Pereira, F. C. Beyond expectation: Deep joint mean and quantile regression for spatiotemporal problems. *IEEE transactions on neural networks and learning systems*, 31(12):5377–5389, 2020.

[Samuel, 1959] Samuel, A. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development*, 3(3):210–229, 1959.

[Sangnier et al., 2016] Sangnier, M., Fercoq, O., and d'Alché Buc, F. Joint quantile regression in vector-valued RKHSs. In *Neural Information Processing Systems*, 2016.

[Sauer, 2011] Sauer, T. *Numerical analysis*. Addison-Wesley Publishing Company, 2011.

[Savine, 2018] Savine, A. *Modern computational finance: AAD and parallel simulations*. John Wiley & Sons, 2018.

[Schönbucher, 2004] Schönbucher, P. A measure of survival. *Risk Magazine*, 17(8):79–85, 2004.

[Shahriari et al., 2015] Shahriari, B., Swersky, K., Wang, Z., Adams, R. P., and De Freitas, N. Taking the human out of the loop: A review of Bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[Shalev-Shwartz and Ben-David, 2014] Shalev-Shwartz, S. and Ben-David, S. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[Shapiro et al., 2014] Shapiro, A., Dentcheva, D., and Ruszczynski, A. *Lectures on Stochastic Programming - Modeling and Theory, Second Edition*. SIAM, 2014.

[Shapiro et al., 2021] Shapiro, A., Dentcheva, D., and Ruszczynski, A. *Lectures on stochastic programming: modeling and theory*. SIAM, 2021.

[Shen et al., 2021] Shen, G., Jiao, Y., Lin, Y., Horowitz, J. L., and Huang, J. Deep Quantile Regression: Mitigating the Curse of Dimensionality Through Composition. *arXiv preprint arXiv:2107.04907*, 2021.

[Shi et al., 2016] Shi, S., Wang, Q., Xu, P., and Chu, X. Benchmarking state-of-the-art deep learning software tools. In *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*, pages 99–104. IEEE, 2016.

[Shorten and Khoshgoftaar, 2019] Shorten, C. and Khoshgoftaar, T. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

[Sidorov et al., 2020] Sidorov, O., Hu, R., Rohrbach, M., and Singh, A. Textcaps: a dataset for image captioning with reading comprehension. In *European Conference on Computer Vision*, pages 742–758. Springer, 2020.

[Sirko et al., 2021] Sirko, W., Kashubin, S., Ritter, M., Annkah, A., Bouchareb, Y. S. E., Dauphin, Y., Keysers, D., Neumann, M., Cisse, M., and Quinn, J. Continental-scale building detection from high resolution satellite imagery. *arXiv preprint arXiv:2107.12283*, 2021.

[Slivkins, 2019] Slivkins, A. Introduction to multi-armed bandits. *arXiv preprint arXiv:1904.07272*, 2019.

[Son et al., 2021] Son, H., Jang, J. W., Han, W. J., and Hwang, H. J. Sobolev Training for Physics Informed Neural Networks. *arXiv preprint arXiv:2101.08932*, 2021. https://arxiv.org/abs/2101.08932.

[Squire and Trapp, 1998] Squire, W. and Trapp, G. Using complex variables to estimate derivatives of real functions. *SIAM review*, 40(1):110–112, 1998.

[Stone et al., 2010] Stone, J. E., Gohara, D., and Shi, G. OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in science & engineering*, 12(3):66, 2010.

[Swiss Federal Office of Private Insurance, 2006] Swiss Federal Office of Private Insurance. Technical document on the Swiss solvency test, 2006. https://www.finma.ch/FinmaArchiv/

bpv/download/e/SST_techDok_061002_E_wo_Li_20070118.pdf.

[Takei, 2020] TAKEI, R. Flattening the (Funding) Curve, 2020. https://ihsmarkit.com/research-analysis/flattening-the-funding-curve.html.

[Takeuchi et al., 2006] TAKEUCHI, I., LE, Q. V., SEARS, T. D., and SMOLA, A. J. Nonparametric Quantile Estimation. *Journal of Machine Learning Research*, 7:1231–1264, 2006. ISSN 1532-4435.

[Teng, 2021] TENG, L. Gradient boosting-based numerical methods for high-dimensional backward stochastic differential equations. *arXiv preprint arXiv:2107.06673*, 2021.

[The Bank for International Settlements, 2011] THE BANK FOR INTERNATIONAL SETTLEMENTS. Capital treatment for bilateral counterparty credit risk finalised by the Basel Committee, 2011. https://www.bis.org/press/p110601.htm.

[Tian et al., 2015] TIAN, Y., ZHU, Z., LEE, G., KLEBANER, F., and HAMZA, K. Calibrating and pricing with a stochastic-local volatility model. *The Journal of Derivatives*, 22(3):21–39, 2015.

[Tsitsiklis and Van Roy, 2001] TSITSIKLIS, J. N. and VAN ROY, B. Regression methods for pricing complex American-style options. *IEEE Transactions on Neural Networks*, 12(4):694–703, 2001.

[Vapnik, 1991] VAPNIK, V. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

[Vershynin, 2018] VERSHYNIN, R. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge university press, 2018.

[Vlassis and Sun, 2021] VLASSIS, N. N. and SUN, W. Sobolev training of thermodynamic-informed neural networks for interpretable elasto-plasticity models with level set hardening. *Computer Methods in Applied Mechanics and Engineering*, 377:113695, 2021.

[Voulodimos et al., 2018] VOULODIMOS, A., DOULAMIS, N., DOULAMIS, A., and PROTOPAPADAKIS, E. Deep learning for computer vision: A brief review. *Computational intelligence and neuroscience*, 2018, 2018.

[Weinan et al., 2019] WEINAN, E., HUTZENTHALER, M., JENTZEN, A., and KRUSE, T. On multilevel Picard numerical approximations for high-dimensional nonlinear parabolic partial differential equations and high-dimensional nonlinear backward stochastic differential equations. *Journal of Scientific Computing*, 79(3):1534–1571, 2019.

[Wolf et al., 2019] WOLF, T., DEBUT, L., SANH, V., CHAUMOND, J., DELANGUE, C., MOI, A., CISTAC, P., RAULT, T., LOUF, R., FUNTOWICZ, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

[Young et al., 2018] YOUNG, T., HAZARIKA, D., PORIA, S., and CAMBRIA, E. Recent trends in deep learning based natural language processing. *ieee Computational intelligenCe magazine*, 13(3):55–75, 2018.

[Zaharia et al., 2016] ZAHARIA, M., XIN, R. S., WENDELL, P., DAS, T., ARMBRUST, M., DAVE, A., MENG, X., ROSEN, J., VENKATARAMAN, S., FRANKLIN, M. J., et al. Apache spark: a unified engine for big data processing. *Communications of the ACM*, 59(11):56–65, 2016.

[Zenati et al., 2018] ZENATI, H., FOO, C. S., LECOUAT, B., MANEK, G., and CHANDRASEKHAR, V. R. Efficient gan-based anomaly detection. *arXiv preprint arXiv:1802.06222*, 2018.

[Zhang, 2004] ZHANG, J. A numerical scheme for BSDEs. *The Annals of Applied Probability*, 14(1):459–488, 2004.

[Zhang et al., 2019] ZHANG, S., YAO, L., SUN, A., and TAY, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38, 2019.

université PARIS-SACLAY | ÉCOLE DOCTORALE de mathématiques Hadamard (EDMH)

Titre : Apprentissage sur données simulées en finance : XVAs, mesures de risque et calibration
Mots clés : apprentissage statistique, finance quantitative, calcul sur GPU, calibration, mesures de risque, XVA

Résumé : L'émergence de cadres XVA complexes et de modèles d'évaluation coûteux en temps de calcul a encouragé les chercheurs et les praticiens de la finance à se pencher sur les méthodes d'apprentissage statistique pour accélérer leurs calculs. Cette thèse vise à proposer de nouvelles approches basées sur les réseaux de neurones. Tout d'abord, nous proposons un cadre XVA cohérent et une implémentation pratique utilisant des régressions par moindres carrés et des régressions quantiles/expected shortfall avec des réseaux de neurones et le calcul sur GPU. Notre implémentation évite les simulations Nested Monte Carlo et n'a pas besoin des approximations habituelles utilisées par les praticiens. Ensuite, nous abordons la question de l'apprentissage des espérances ou des mesures de risque conditionnelles en présence d'événements de défaut dans un cadre général. Nous proposons pour cela un nouveau schéma de simulation et fournissons une analyse de convergence statistique et des expériences numériques démontrant son efficacité. Nous étudions également la convergence statistique d'une approche d'apprentissage de quantile et expected shortfall en deux étapes et nous proposons des schémas d'apprentissage basés sur des réseaux de neurones pour les cas à un et plusieurs quantiles. Nous abordons aussi la question du croisement des quantiles. Motivés par le fait que la fongibilité du capital à risque avec la marge de variation dans les calculs XVA donne lieu à des équations différentielles stochastiques rétrogrades anticipées, nous proposons un schéma d'apprentissage explicite pour de telles équations. Enfin, nous proposons une approche de projection pour approximer le prix des options vanilles dans un contexte de calibration de modèles pour accélérer cette dernière. Notre méthode, basée sur la différenciation à pas complexe, enrichit l'apprentissage en cherchant à projeter des dérivées directionnelles stochastiques.

Title : Learning From Simulated Data in Finance : XVAs, Risk Measures and Calibration
Keywords : statistical learning, quantitative finance, GPU computing, calibration, risk measures, XVA

Abstract : The emergence of complex XVA frameworks and time-consuming pricing models has encouraged researchers and finance practitioners to look at statistical learning methods to accelerate their calculations. The present thesis aims to contribute new approaches based on neural networks. First, we propose a consistent XVA framework along with a practical implementation using neural networks least-squares and quantile/expected shortfall regressions and GPU computing. Our implementation avoids Nested Monte Carlo simulations and does not need the usual approximations used by practitioners. Then, we address the issue of learning conditional expectations or risk measures in the presence of default events in a general framework. For this, we propose a new simulation scheme and provide a statistical convergence analysis and numerical experiments demonstrating its effectiveness. We also study the statistical convergence of a two-step quantile and expected shortfall learning approach and provide learning schemes based on neural networks for the single and multiple quantile learning cases. We address the quantile crossing issue as well. Motivated by the fact that the fungibility of the risk capital with variation margin in XVA calculations gives rise to anticipated backward stochastic differential equations, we devise an explicit learning scheme for such equations. Finally, we provide a projection approach to approximate the price of vanilla options in the context of model calibration to accelerate the latter. Our method, based on complex-step differentiation, augments the learning by seeking to project stochastic directional derivatives.