A machine learning approach to portfolio pricing and risk management for high-dimensional problems

Lucio Fernandez-Arjona¹ | Damir Filipović²

1University of Zurich, Zurich, Switzerland
2EPFL and Swiss Finance Institute, Lausanne, Switzerland

Correspondence
Damir Filipović, EPFL and Swiss Finance Institute, Lausanne, Switzerland.
Email: damir.filipovic@epfl.ch

Abstract
We present a general framework for portfolio risk management in discrete time, based on a replicating martingale. This martingale is learned from a finite sample in a supervised setting. Our method learns the features necessary for an effective low-dimensional representation, overcoming the curse of dimensionality common to function approximation in high-dimensional spaces, and applies for a wide range of model distributions. We show numerical results based on polynomial and neural network bases applied to high-dimensional Gaussian models. In these examples, both bases offer superior results to naive Monte Carlo methods and regress-now least-squares Monte Carlo (LSMC).

KEYWORDS
dimensionality reduction, nested Monte Carlo, neural networks, replicating portfolios, solvency capital

This is an open access article under the terms of the Creative Commons Attribution-NonCommercial-NoDerives License, which permits use and distribution in any medium, provided the original work is properly cited, the use is non-commercial and no modifications or adaptations are made.

© 2022 The Authors. Mathematical Finance published by Wiley Periodicals LLC.
1 | INTRODUCTION

Financial institutions face a variety of risks on their portfolios. Whether they be market and credit risk for investment portfolios, default, and prepayment risk on their mortgage portfolios, or longevity risk on life insurance portfolios, the balance sheet of a bank or insurance company is exposed to many risk factors. Failure to manage these risks can lead to insolvency—with associated losses to shareholders, bondholders and/or customers—or overly conservative business strategies, which hurt consumers.

Alongside qualitative assessments—and plenty of common sense—portfolio risk management requires quantitative models that are accurate and sufficiently fast to provide useful information to management. Additionally, government regulations, such as solvency regimes, require extensive calculations to produce the required reports. Simulation techniques are often used to explore possible future outcomes. Since quantitative models require to estimate conditional expectations across a time interval, Monte Carlo simulations can be used to calculate those expectations. However, plain Monte Carlo methods suffer from problems with both accuracy and speed.

Alternative methods have been developed over the years, some of them using functional approximation techniques. Under this approach, an approximation to the full, slow model is built using one or more faster functions. For example, the behavior of a portfolio can be replicated via an appropriate combination of basis functions, which are faster to calculate than the original model. Most—if not all—of these alternatives suffer from several problems. Some of them are not data-driven—requiring subject matter expertise—which limits their applicability to complex problems and the ability to automate them. Others can be automated but have low quality of the approximation. And some other again are limited to low-dimensional problems.

In this paper, we present a method that overcomes or greatly diminishes these problems. Our method for calculating conditional expectations uses a machine learning approach to learn a suitable function from finite samples. Therefore, the entire process is data-driven and can be free of manual steps. We show how this function can be used to obtain accurate estimates of price and risk measures, focusing on practical real-world situations in terms of runtime and number of samples being used.

The learned functions are linear combinations of functional bases, of which we present several examples, including polynomials and neural networks (of the single-layer feed-forward type). In all cases, the conditional expectations are calculated in closed-form—even for neural networks—which contributes to the accuracy and speed of the solution. We aim at high-dimensional cases, where working with a full polynomial basis is unfeasible due to the combinatorial explosion of the number of basis functions. This motivates the use of a special polynomial basis. In this basis, the input vector undergoes a data-driven, linear dimensionality reduction (LDR) step—similar to a linear neural network layer—while remaining tractable with closed-form solutions, as we show. While our numerical examples are based on Gaussian measures, the method applies to a wide range of model distributions.

In probabilistic terms, we obtain martingales that replicate the value processes—given as risk-neutral conditional expectations—of financial or insurance products. Drawing a parallel with the concept of a replicating portfolio, we also call our approach the replicating martingale method. Replicating portfolios is a widely used method in the financial industry, which relies on building linear combinations of derivatives to approximate conditional expectations necessary for market risk calculations. Our proposed replicating martingale method is also based on linear combinations of basis functions, but these are not restricted to market risk calculations. Any risk exposure
can be modeled with replicating martingales because its basis functions—polynomials or neural networks—are agnostic to the underlying risk type.

Given its regression-based nature and use of simulated samples, replicating martingales is a method of the least squares Monte Carlo (LSMC) family. Within the LSMC family of methods, replicating martingales are a regress-later method (Glasserman & Yu, 2002) given the regression is made against terminal payoffs and not against empirical conditional expectations.

We implement two numerical examples of high-dimensional Gaussian models—a plain European call option and a path-dependent life insurance product—to perform extensive testing of the accuracy of the risk calculations based on replicating martingales. In line with existing machine learning literature, we also include extensive comparisons with alternative methods, such as nested Monte Carlo (nMC) and other LSMC methods, and we find our method offers superior results.

1.1 | Related literature

An early example of static replication via basis functions can be found in Madan and Milne (1994), which presents a framework for replication of general contingent claims. These contingent claims are modeled in a Hilbert space and the static replication problem is solved by constructing a countable orthonormal basis. The method is applied to the pricing and hedging of these contingent claims. Carriere (1996) and Longstaff and Schwartz (2001) use a sequential approximation algorithm to calculate the conditional expectations required in the valuation of options with an early exercise (like American options). The method estimates these conditional expectations from the cross-sectional information in the simulation by using least squares, which gives them the method the name of LSMC. Andreattà and Corradin (2003) apply this idea to valuation of life insurance policies.

The first distinction between regress-now and regress-later LSMC appears in Glasserman and Yu (2002). The former is the direct estimation of the conditional expectation function, and the latter the indirect estimation via regression against the terminal payoff of the contingent claim. Working in the context of American option pricing via approximate dynamic programming, they find that regress-later LSMC give less-dispersed estimates than regress-now LSMC.

The distinction and relationship between regress-now and regress-later models are important to understand our method. Regress-later models produce—ceteris paribus—better approximation functions, but introduce a few difficulties, among them the need to solve a much larger regression problem and the need to evaluate the conditional expectation of the approximation function. At the core of our paper is the demonstration of how—for polynomials and neural networks as bases—one can overcome the problem of higher dimensionality using LDR, and one can calculate conditional expectations in closed form. These two factors compensate the difficulties introduced by the regress-later approach, and allow to achieve better results than a comparable regress-now approach would.

In the area of polynomial regress-now LSMC, Broadie et al. (2015) show that such regression-based methods can—asymptotically—improve the convergence rate of nMC methods. They provide quality comparisons against nMC, and a delta-gamma approach but not against regress-later methods, whereas we do.

A regress-later model based on orthonormal piecewise linear basis functions is presented in Pelsser and Schweizer (2016). Path-dependency and the resulting high-dimensional problem in long-term projections is managed via a hand-picked dimensionality-reduction function to avoid
the curse of dimensionality. In that framework, the basis functions are not guaranteed to have a 
closed form solution, whose existence depends on the choice of dimensionality-reduction func-
tion. This function must be given to the method, and is based on expert judgment and knowledge 
of the problem domain. Moreover, the choice of this function implies a trade-off between com-
plexity and dimensionality—for a given target accuracy. High-dimensional functions lead to the 
curse of dimensionality while low-dimensional functions might be too complex to find a closed-
form solution to their conditional expectations. By contrast, our framework uses a data-driven 
dimensionality-reduction function in the parameter space instead of an arbitrary function. In 
comparison to the piecewise linear model, which requires fixing a grid, neural networks are able 
to provide a data-driven grid for its activation functions.

Another application of piecewise polynomials can be found in Duong (2019). In this case, the 
method is based on splines in a regress-now setting. In contrast, our models are based on global 
polynomials in a regress-later setting.

A neural network model is applied to solvency capital problem in life insurance in Castel-
lani et al. (2018). The neural network model shows better performance than LSMC—regress-now 
model with polynomial basis functions—and a support vector regression model. All three 
models—including the neural network model—are regress-now models. By contrast, we focus 
on regress-later methods, which show better accuracy in the examples in addition to being better 
in theory.

Another approach for the calculation of risk metrics using a functional approximation 
in presented in Ha and Bauer (2020), which presents a LSMC regress-now method with a 
data-driven selection of basis functions, with an example given for a Gaussian model and 
Hermite polynomials.

The literature also contains other examples of methods not based on polynomials or neural net-
works, for example Hong et al. (2017) and Risk and Ludkovski (2018). Those papers show methods 
to approach the same problem as the replicating martingales in this paper, namely the calculation 
of risk metrics for risk management purposes, but using kernel methods and Gaussian process 
regression, respectively.

Table 1 summarizes the above discussion and gives an overview of the related literature on 
regress-now and regress-later methods.

Another strand of related literature is in the field of uncertainty quantification, where polyno-
mial surrogate functions have been used for a long time to reduce the runtime of complex models. 
Recently, Hokanson and Constantine (2018) showed that polynomial ridge approximation can be 
extended to reduce dimensionality in a data-driven manner. We follow a similar approach and 
apply it to portfolio pricing and risk management.

nMC methods for portfolio management have been studied in Lee and Glynn (2003) and Gordy 
and Juneja (2010) among others. Gordy and Juneja (2010) show several methods that can reduce 
the computational cost of nMC for a homogeneous portfolio of financial instruments with an 
additive structure. Since we aim to cover portfolios do not exhibit such additive structure—such 
as the life insurance portfolio—we restrict our comparisons to standard nMC.

An unfortunate aspect of this literature is that there are no standard models on which to com-
pare the quality of the surrogate models, as is the case in the field of machine learning with 
datasets like MNIST LeCun et al. (1998). Such a dataset would allow more straightforward com-
parisons among the advanced methods mentioned above. As in the large majority of the above 
literature, we compare the replicating martingale approach to some well-established methods— 
in our case nMC and regress-now LSMC—but not to any of the other advanced methods proposed 
in the literature. In this sense, we do not claim any general superiority of our approach.
TABLE 1  Comparison of regress-now and regress-later methods in the literature. LDR stands for linear dimensionality reduction

|                        | Dimensionality reduction on inputs | Regression method                              |
|------------------------|------------------------------------|------------------------------------------------|
| **Regress-now**         |                                    |                                                 |
| Broadie et al. (2015)   | None                               | Incomplete, manually selected monomial basis    |
| Duong (2019)            | None                               | Polynomial spline space                        |
| Castellani et al. (2018)| Implicit in neural network         | Neural network                                 |
| Ha and Bauer (2020)     | None                               | Left singular functions of the conditional expectation operator |
| Hong et al. (2017)      | Model decomposition                | Kernel smoothing                               |
| Risk and Ludkovski (2018)| None                             | Gaussian process regression                    |
| **Regress-later**       |                                    |                                                 |
| Pelsser and Schweizer (2016)| Manually selected function           | Orthonormal piecewise linear basis and Hermite polynomials |
| this paper              | LDR                                | orthogonal polynomials and shallow neural networks |

The remainder of the paper is as follows. Section 2 formalizes the replicating martingale problem and recalls the standard nMC approach, which is then illustrated by means of a preliminary example in Section 3. Section 4 describes our machine learning approach to the replicating martingale problem. It contains novel, rigorous results on the existence and uniqueness of the optimal surrogate function. Sections 5 and 6 provide numerical case studies: the initial European call option example revisited in Section 5, and an insurance liability model in Section 6. Section 7 concludes. The appendix contains proofs and technical background material. Appendix A describes the quality metrics used to compare different methods. Appendix B contains the economic scenario generator underlying the numerical examples in the main text. Appendix C contains all proofs and some auxiliary results of independent interest. A comparison of the runtimes of the methods is given in Appendix D. Appendix E presents an analysis of the sensitivity of the proposed methods to different hyper-parameters.

2  THE REPLICATING MARTINGALE PROBLEM

We consider an economic scenario generator with a finite time horizon $T$, where time is in units of years. Randomness is generated by an $\mathbb{R}^d$-valued stochastic driver process $X = (X_1, \ldots, X_T)$ with mutually independent components $X_i$. We denote by $\mathbb{Q}$ the distribution of $X$ on the path space $\Omega = \mathbb{R}^{dT}$. The flow of information is modeled by the filtration $\mathcal{F}_t = \sigma(X_1, \ldots, X_t), t = 1, \ldots, T$, generated by $X$. If not otherwise stated, all financial values and cash flows are discounted by some numeraire, for example, the cash account, and we assume that $\mathbb{Q}$ is the corresponding risk-neutral pricing measure.
Our objective is a portfolio of assets and liabilities whose present value is to be derived from its cash flow, which accumulates to a terminal value at $T$ given as function $f$ of $X$,

$$f(X) = \sum_{t=1}^{T} \zeta_t,$$  \hspace{1cm} (1)

with $P_t$-measurable time-$t$ cash flows $\zeta_t = \zeta_t(X_1, \ldots, X_t)$. As is the case in practice, we assume that all $\zeta_t$, and thus $f$, are exogenously given functions in $L^2_{\mathbb{Q}}$. Our goal is to find the cum-dividend value process of the portfolio, given as

$$V_t = \mathbb{E}^Q_t[f(X)] = \sum_{s=1}^{t} \zeta_s \text{ accumulated cash flow at } t + \mathbb{E}^Q_t \left[ \sum_{s=t+1}^{T} \zeta_s \text{ time-}t \text{ spot value} \right],$$  \hspace{1cm} (2)

where $\mathbb{E}^Q_t[\cdot] = \mathbb{E}[\cdot | P_t]$ denotes the $P_t$-conditional expectation. Formally speaking, $V_t$ is the $L^2_{\mathbb{Q}}$ martingale that replicates the terminal value $f(X)$.

There are many examples that fit this description, including complex financial derivatives, insurance liabilities, mortgage-backed instruments, and other structured products. In most real-world cases, $V_t$ is not given in closed form but has to be estimated from simulating $f(X)$. This creates computational challenges, as the function $f$ may be costly to query and available computational budget is limited.

There are several risk management applications of the portfolio value process. We focus here on risk measurement. Insurance and banking solvency regulatory frameworks, such as Solvency II, Swiss Solvency Test, and Basel III, require capital calculations that are based on risk measurements of the value changes $\Delta V_t = V_t - V_{t-1}$ over risk periods $(t - 1, t]$. The most common risk measures are value at risk $\text{VaR}_\alpha(L)$, defined as the left $\alpha$-quantile of the distribution of the loss $L$, for some confidence level $\alpha \in (0, 1)$, and expected shortfall $\text{ES}_\alpha(L) = \frac{1}{1-\alpha} \int_{-\infty}^{1-\alpha} \text{VaR}_\gamma(L) d\gamma$, see, for example, (Föllmer & Schied, 2004, Section 4.4). In insurance regulation, risk is usually measured for a 1-year risk horizon and the economic capital is determined by $\rho[-\Delta V_1]$, where $\rho$ is a placeholder for either $\text{VaR}_\alpha$ or $\text{ES}_\alpha$. There are cases, however, where calculations require 3-year capital projections, which would be calculated by $\rho[-\Delta V_1], \rho[-\Delta V_2]$ and $\rho[-\Delta V_3]$ for years one, two and three, respectively, which would require the joint distribution of (parts of) the entire process $V$.

Monte Carlo simulation is the standard method for computing risk measures. We refer to as standard nMC to the method that is illustrated in Figure 1. The outer stage of the simulation consists of a set of $n_0$ simulations, $X_{1:t}^{(i)}, i = 1, \ldots, n_0$, that are independent and identically distributed as the stochastic driver $X$ up to the risk horizon time $t$. The portfolio value $V_t^{(i)}$ for each outer scenario $X_{1:t}^{(i)}$ is estimated via the inner stage of Monte Carlo simulation by taking the sample mean of the $n_1$ inner simulations drawn for each outer simulation $X_{1:t}^{(i)}$. Once the empirical distribution for $V_t^{(i)}$ has been obtained, the desired risk measure is approximated via its empirical equivalent. While straightforward to implement, standard nMC leads to an exponential increase of simulations if applied to various risk time horizons. Hence, it often cannot be in used in practice due to the large computational effort required. We illustrate these issues with the following initial example of a European call option.
FIGURE 1 Nested Monte Carlo structure. The outer simulations span the time period 0 to \( t \), and each of those serves as the starting point of a group of inner simulations [Color figure can be viewed at wileyonlinelibrary.com]

3 | NESTED MONTE CARLO FOR A EUROPEAN CALL OPTION

By way of preliminary example, we calculate the present value \( V_0 \) and expected shortfall of the 1-year loss \(-\Delta V_1\) for a European call on an equity index. We assume that we hold a short position in this call and therefore, focus on the loss-making tail, that is, the tail where the equity index values are higher.

The economic scenario generator is described in Appendix B. That generator maps \( X \) to a vector of economic factors. This vector contains several components, among them the equity index \( S_t \) and the cash account \( C_t \). The European call payoff at time \( T \) is \( \max(S_T - K, 0) \) where \( K \) is the strike of the option and \( T \) its maturity. The variables \( S_t, C_t \), and \( K \) represent nominal—undiscounted—values. Hence, for this portfolio, there is one discounted cash flow at \( T \), so that the terminal value function is

\[
f(X) = -\frac{\max(S_T - K, 0)}{C_T}.
\]

(3)

In this example, we work with two maturities \( T = 5 \) and \( T = 40 \) and for both strike price \( K = S_0 = 100 \). The model has three stochastic drivers, \( d = 3 \), and therefore, there are 15 and 120 total dimensions—individual stochastic variables \( X \)—for \( T = 5 \) and \( T = 40 \), respectively.

To establish the benchmark value—the closest we can get to the ground truth without using a closed form solution\(^1\)—we first run a very large nMC simulation, with 1000,000 outer simulations \((n_0)\) and 100,000 inner simulations \((n_1)\). We then calculate the 99% expected shortfall on the loss-making tail. When working with an empirical distribution, as we do here based on simulation data, the expected shortfall reduces to simply averaging the 1% worst results.

Before we can test the quality of the nMC estimator for a finite simulation budget, we need to decide how to split said budget between inner and outer simulations. Different combinations of outer and inner simulations will produce different nMC estimators. The bias and variance of the nMC estimator depends on both the amount of outer and inner simulations. Each combination has a different bias and variance and therefore, a different mean absolute error. This is shown in
Table 2 for a fixed total budget of $n_0 \times n_1 = 50,000$ simulations. In each individual estimation (inner–outer combination), the error is calculated as a percentage of the benchmark value and therefore, we refer to the quality metric as MApE, for mean absolute percentage error. The formula is described in Appendix A. It is important to note that since the expected shortfall is calculated on $-\Delta V_1$, the error on $V_0$ is also part of the error on $\text{ES}_{99\%}(-\Delta V_1)$.

As described in Broadie et al. (2015), it is not possible, in general cases, to decide for a finite budget how to make this inner-out split in an optimal way. In this paper, we will err on the side of presenting optimistic risk figures for nMC estimations, by choosing an optimal split corresponding to the smallest MApE ES. This bias towards more accurate nMC risk estimations than possible in practice will not be a problem for our analysis, since we find that the proposed replicating martingale method produces more accurate results than the optimal nMC, which is already better than what one would obtain in practice. Note that this problem does not exist for regression-based methods, since it is not necessary to split the training budget.

Table 3 shows the MApE for present value and ES of the optimal combination for varying total sample size $n_0 \times n_1$. We can see that the estimation of $\text{ES}_{99\%}(-\Delta V_1)$, performed via nMC, is much more affected by the sample size than the estimation of $V_0$. In fact, for the estimation of $V_0$, it would be optimal not to split the simulations. For example, the MApE for the present value for sample size 50,000 would be 0.6% and 1%, as opposed to 1.7% and 2.4% as shown in the last row, respectively.

These Monte Carlo results will be used throughout the paper as one of the reference methods against, which we measure our approach. Even if the limitations of standard nMC mean that it is not the main approach used by practitioners in large-scale problems, it remains the simplest way to approach the estimation of conditional expectations, and it provides a common baseline that both practitioners and academics can easily understand.

4 | MACHINE LEARNING APPROACH

We now present our method, which addresses the computational challenges described and illustrated by the above example. Thereto, in a first step, we directly approximate the terminal
value function $f$ by projecting it on a finite-dimensional subspace in $L^2_\mathbb{Q}$, that is spanned by an optimally chosen set of basis functions in $X$, which constitute the feature map. We assume that these basis functions admit conditional expectations in closed form. We thus obtain, as second step, an approximation of the portfolio value process in closed form. In practical applications, we learn the approximation of $f$ from a finite sample of $X$, which induces an empirical measure that proxies the model population measure $\mathbb{Q}$. The performance of this approach hinges on the choice of the basis functions. We formalize and discuss all this in detail in the following.

4.1 \hspace{1cm} \textbf{Finite-dimensional approximation}

Fix a dimension $m \in \mathbb{N}$, and let $\Theta$ be a parameter set such that, for every $\theta \in \Theta$, there are functions $\phi_{\theta,i} : \mathbb{R}^{dT} \to \mathbb{R}$ in $L^2_\mathbb{Q}$, for $i = 1, \ldots, m$. These functions form the feature map $\phi_\theta = (\phi_{\theta,1}, \ldots, \phi_{\theta,m})^T$. For any $\theta \in \Theta$, the $L^2_\mathbb{Q}$-projection of $f$ on span{$\phi_{\theta,1}, \ldots, \phi_{\theta,m}$} is given by $f_\theta = \sum_{i=1}^{m} \phi_{\theta,i}\beta_{\theta,i} = \phi_\theta^T\beta_\theta$, where $\beta = \beta_\theta \in \mathbb{R}^m$ solves

$$
\min_{\beta \in \mathbb{R}^m} \|f - \phi_\theta^T\beta\|_{L^2_\mathbb{Q}}. \tag{4}
$$

We assume that the conditional expectations $\mathbb{E}^\mathbb{Q}_t[\phi_\theta(X)] = G_{\theta,t}(X_1, \ldots, X_t)$ are given in closed form, in the sense that the conditional expectation functions $G_{\theta,t} : \mathbb{R}^{dT} \to \mathbb{R}^m$, given by

$$
G_{\theta,t}(x_1, \ldots, x_t) = \mathbb{E}^\mathbb{Q}[\phi_\theta(x_1, \ldots, x_t, X_{t+1}, \ldots, X_T)], \tag{5}
$$

can be efficiently evaluated at very low computational cost. As a result, we obtain the approximate value process

$$
V_{\theta,t} = G_{\theta,t}(X_1, \ldots, X_t)^T\beta_\theta, \tag{6}
$$
in closed form. This requirement is a key distinction of our method from other methods in the literature. We obtain the value process of the portfolio by regressing against its terminal value. Our approach, therefore, falls within the regress later category first mentioned in Glasserman and Yu (2002). As we will see in the numerical examples below, this approach performs better than the alternative regress now.

So, how good is our approximation (6)? By Doob’s inequality, making use of the martingale property of the value process, we obtain an upper bound on the pathwise maximal $L^1_\mathbb{P}$-error,

$$
\max_{t \leq T} |V_t - V_{\theta,t}|_{L^1_\mathbb{P}} \leq \frac{d\mathbb{P}}{d\mathbb{Q}} \|f - f_\theta\|_{L^2_\mathbb{Q}} \leq 2 \frac{d\mathbb{P}}{d\mathbb{Q}} \|f - f_\theta\|_{L^2_\mathbb{Q}}. \tag{7}
$$

Note that $\|\frac{d\mathbb{P}}{d\mathbb{Q}}\|_{L^2_\mathbb{Q}}$ is known by the modeler. The $L^2_\mathbb{Q}$-approximation error on the right hand side of Equation (7) is the objective in Equation (4). In practice, it can be estimated by Monte Carlo based on the training sample used to learn $f_\theta$. Hence, albeit elementary, this inequality gives a practical upper bound on the relevant $L^1_\mathbb{P}$-error of the approximation of $V_t$. 
4.2 Feature learning

We now learn the parameter $\theta$ from the data. This is a second key distinction of our method, which allows to tackle the notorious curse of dimensionality that comes with polynomial feature maps, as we shall see below. Thereto we minimize the approximation error by an optimal choice of $\theta$, which leads to the nonconvex optimization problem

$$
\min_{(\theta, \beta) \in \Theta \times \mathbb{R}^m} \| f - \phi_\theta^T \beta \|_{L_Q^2}.
$$

(8)

Here is an elementary existence result.

**Lemma 4.1.** Assume

(i) $\Theta$ is compact,

(ii) $\theta \mapsto \phi_{\theta,i} : \Theta \to L_Q^2$ is continuous for all $i$,

(iii) $\{\phi_{\theta,1}, \ldots, \phi_{\theta,m}\}$ is linearly independent in $L_Q^2$ for all $\theta \in \Theta$.

Then there exists a solution to Equation (8).

The assumptions in Lemma 4.1 cannot be relaxed in general. This is shown by the following example.

**Example 4.2.** Let $Q = \frac{1}{2} \sum_{i=1}^2 \delta_{x(i)}$ be a discrete measure supported on two points $x(1) \neq x(2)$, so that every function $f \in L_Q^2$ can be identified with the $\mathbb{R}^2$-vector $(f(x(1)), f(x(2)))$. We let $f = (1,0)$, $m = 1$, and either

(i) $\Theta = (0,1]$ (not compact) and $\phi_\theta = \frac{(1,\theta)}{\sqrt{1+\theta^2}}$, or

(ii) $\Theta = [0,1]$ and $\phi_\theta = \frac{(1,0,1)(\theta), \phi_{10(\theta)}}{\sqrt{1+\theta^2}}$, so that $\phi_0 = (0,1)$ (not continuous), or

(iii) $\Theta = [0,1]$ and $\phi_\theta = \phi_{(1,\theta)} \frac{(0,1)}{\sqrt{1+\theta^2}}$, so that $\phi_0 = (0,0)$ (not linearly independent).

For either case, we have $\inf_{(\theta, \beta) \in \Theta \times \mathbb{R}} \| f - \phi_\theta \beta \|_{\mathbb{R}^2} = \lim_{\theta \to 0} \| f - \phi_\theta \beta \|_{\mathbb{R}^2} = 0$, but the infimum is not attained, $\| f - \phi_\theta \beta \|_{\mathbb{R}^2} > 0$ for all $(\theta, \beta) \in \Theta \times \mathbb{R}$.

In practice, there are many factors that determine whether the approximation (8) will be close to the true $f$. One of them is the relationship between the dimension of the random driver, $dT$, and the size of the training sample. Since in real-world applications, the sample size is limited by practical constraints, it is necessary to reduce the effective dimension of the stochastic driver. This motivates the use of a LDR, as follows. We henceforth assume that the feature map is parametrized in the form

$$
\phi_{\theta,i}(x) = g_i(A^T x + b),
$$

(9)
for some exogenously given functions $g_i : \mathbb{R}^p \to \mathbb{R}$, $i = 1, \ldots, m$, for some $p \in \mathbb{N}$, and the parameter $\theta = (A, b)$ consists of a weight matrix $A$ and a bias vector $b$, for a subset $\Theta \subseteq \mathbb{R}^{dT \times p} \times \mathbb{R}^p$. 

As a notational convention, to capture the evaluation of $\phi_{\theta}$ at the $\mathbb{R}^{dT}$-valued stochastic driver $X = (X_1, \ldots, X_T)$, we decompose the $dT \times p$-matrix $A$ into $T$ consecutive blocks $A_t \in \mathbb{R}^{d \times p}$ such that $A^T = (A_1^T, \ldots, A_T^T)$. Then we have $A^T \text{vec}(X) = \sum_{t=1}^T A_t^T X_t$ and

$$\phi_{\theta,i}(X) \equiv \phi_{\theta,i}(\text{vec}(X)) = g_i(\sum_{t=1}^T A_t^T X_t + b). \quad (10)$$

The conditional expectation functions in Equation (5) read component-wise for $i = 1, \ldots, m$ as

$$G_{\theta,i}(x_1, \ldots, x_t) = \mathbb{E}_\mathbb{Q}[g_i(b + \sum_{s=1}^t A_s^T x_s + \sum_{s=t+1}^T A_s^T X_s)]. \quad (11)$$

We remark that our approach does not replace the original stochastic driver $X$ by $A^T X + b$ in general, only for the specific portfolio $f$ to which it is calibrated through Equation (8). In the sequel, we study three different specifications of the type (9) in more detail: a full polynomial basis, a polynomial feature map with LDR, and a shallow neural network.

### 4.3 Full polynomial basis

We start with the nonweighted and nonbiased case. We formally let $\Theta = \{(I_{dT}, 0)\}$ be the singleton consisting of the $dT \times dT$-identity matrix $A = I_{dT}$ and zero bias vector $b = 0$. Accordingly, we omit the parameter and write shorthand $\phi_{(I_{dT}, 0)} \equiv \phi$. Problem (8) boils down to the projection (4). We let the feature map $\phi$ be composed of a basis of the space of all polynomials of degree $\delta$ or less,

$$\text{Pol}_\delta(\mathbb{R}^{dT}) = \text{span}\{x^\alpha \mid \alpha \in \mathbb{N}^{dT}_0, |\alpha| \leq \delta\}. \quad (12)$$

In order that $\phi_i \in L^2_{\mathbb{Q}}$, we assume that

$$\mathbb{E}_\mathbb{Q}[\|X\|^{2\delta}] < \infty. \quad (13)$$

No attempt is made at selecting individual basis elements from within $\text{Pol}_\delta(\mathbb{R}^{dT})$ at this stage, every polynomial is used in the projection. As a consequence—albeit leading to closed form expressions in Equation (16) below—this feature map suffers the curse of dimensionality from the rapid growth of the number of basis functions $m$ as a function of $d$ and $T$,

$$m = \text{dim Pol}_\delta(\mathbb{R}^{dT}) = \binom{dT+\delta}{dT}. \quad (14)$$

Table 4 shows that the dimension $m$ quickly becomes larger than the training sample size in practice.\(^{3}\)

| $T$ | $d = 3$ | $d = 5$ |
|-----|---------|---------|
| 5   | 816     | 3276    |
| 40  | 302,621 | 1373,701|
The exact form of the conditional expectation (5) depends on the choice of \( \phi \) and the distribution \( \mathcal{Q} = \bigotimes_{t=1}^{T} \mathcal{Q}_t \) of \( X \). Choosing an orthogonal basis of polynomials in \( L^2_{\mathcal{Q}} = \bigotimes_{t=1}^{T} L^2_{\mathcal{Q}_t} \) can greatly simplify the calculations. In view of Sullivan (2015, Theorem 8.25), there is a system of orthogonal polynomials \( \{ g_\alpha \mid \alpha \in \mathbb{N}_0^{dT} \} \) on \( \mathbb{R}^{dT} \) for \( \mathcal{Q} \) that can be expressed as

\[
g_\alpha(x) = \prod_{t=1}^{T} h_{t,\alpha_t}(x_t), \quad x = \text{vec}(x_1, \ldots, x_T), \quad \alpha = (\alpha_1, \ldots, \alpha_T),
\]

where \( \{ h_{t,\alpha_t} \mid \alpha_t \in \mathbb{N}_0^d \} \) is a system of orthogonal polynomials on \( \mathbb{R}^d \) for \( \mathcal{Q}_t \), for every \( t = 1, \ldots, T \). Moreover, \( \deg g_\alpha = |\alpha| \) and \( \deg h_{t,\alpha_t} = |\alpha_t| \). Now choose an index mapping \( \{1, \ldots, m\} \ni i \mapsto \alpha_i \in \{ \alpha \in \mathbb{N}_0^{dT} \mid |\alpha| \leq \delta \} \), then we obtain an orthogonal basis of \( \text{Pol}_\delta(\mathbb{R}^{dT}) \) for \( \mathcal{Q} \) by setting \( \phi_i = g_{\alpha_i} \). The conditional expectation functions (11), where we omit \( \theta \), are then by orthogonality of \( h_{s,\alpha'_s} \) given in closed form as

\[
G_{t,i}(x_1, \ldots, x_t) = \prod_{s=1}^{t} h_{s,\alpha'_s}(x_s) \prod_{s=t+1}^{T} 1_{\alpha'_s=0}.
\]

This extends to the unconditional expectations, \( \mathbb{E}_{\mathcal{Q}}[\phi_i(X)] = \phi_i(0)1_{\alpha'_i=0} \).

**Example 4.3.** Consider the multinormal case \( \text{vec}(X) \sim \mathcal{N}(0, I_{dT}) \). Here we can choose \( g_\alpha \) and \( h_{t,\alpha} \equiv h_\alpha \), as the multivariate (probabilists’) Hermite polynomials of order \( \alpha \in \mathbb{N}_0^{dT} \) on \( \mathbb{R}^{dT} \), and order \( \alpha \in \mathbb{N}_0^d \) on \( \mathbb{R}^d \), respectively.

### 4.4 Polynomial feature map with linear dimensionality reduction

We now tackle the curse of dimensionality of the above full polynomial basis. Thereto we let \( p \leq dT \) and \( \{ g_1, \ldots, g_m \} \) be a basis of \( \text{Pol}_\delta(\mathbb{R}^p) \), and we consider all feature maps (9) for weight matrices \( A \) with full rank and bias vectors \( b \). As above, we assume that Equation (13) holds, so that \( \phi_i(A, b) \in L^2_{\mathcal{Q}} \). The following theorem shows that we can assume that \( b = 0 \) and \( A \) lies in the Stiefel manifold \( V_p(\mathbb{R}^{dT}) = \{ A \in \mathbb{R}^{dT \times p} \mid A^\top A = I_p \} \), the set of all orthonormal \( p \)-frames in \( \mathbb{R}^{dT} \).

**Theorem 4.4.** For any \( A, \tilde{A} \in \mathbb{R}^{dT \times p} \) with full rank and \( b \in \mathbb{R}^p \), the following are equivalent:

(i) \( \tilde{A} \in V_p(\mathbb{R}^{dT}) \) and \( \text{span}\{ \phi_i(A, b), 1, \ldots, \phi_i(A, b), m \} = \text{span}\{ \phi_i(\tilde{A}, 0), 1, \ldots, \phi_i(\tilde{A}, 0), m \} \),

(ii) \( \tilde{A} = A(A^\top A)^{-1/2}U \) for some orthogonal \( p \times p \)-matrix \( U \).

In view of Theorem 4.4, the parameter set can be chosen to be \( \Theta = V_p(\mathbb{R}^{dT}) \) yielding feature maps of the form

\[
\phi_{i}(A, 0)(x) \equiv \phi_{A,i}(x) = g_i(A^\top x),
\]

without loss of generality. We arrive at the following existence and nonuniqueness result.
Table 5: Total dimension \( \dim V_p(\mathbb{R}^d) + \dim \text{Pol}_\delta(\mathbb{R}^p) \) for \( \delta = 3 \)

| \( T \) | \( d = 3, p = 3 \) | \( d = 5, p = 10 \) |
|---|---|---|
| 5 | 9 + 20 = 29 | 195 + 286 = 481 |
| 40 | 114 + 20 = 134 | 1,945 + 286 = 2231 |

Theorem 4.5. For the polynomial feature map, there exists a minimizer in \( \Theta = V_p(\mathbb{R}^d) \) of Equation (8). However, uniqueness does not hold, in the sense that the optimal subspace \( \text{span}\{\phi_{A,1}, \ldots, \phi_{A,m}\} \) is not unique, in general.

Problem (8) corresponds to a LDR with matrix manifold \( V_p(\mathbb{R}^d) \times \mathbb{R}^m \) in the spirit of Cunningham and Ghahramani (2015). The dimensionality reduction is produced exclusively by the linear mapping \( A \) of \( \mathbb{R}^d \) onto \( \mathbb{R}^p \). There is no other restriction imposed on \( \text{Pol}_\delta(\mathbb{R}^p) \), every polynomial basis function \( g_i \) is used. And yet, the dimensionality reduction compared to the full polynomial basis of \( \text{Pol}_\delta(\mathbb{R}^d) \) is significant. Indeed, the total dimension of the optimization problem (8) is given by the sum of \( \dim V_p(\mathbb{R}^d) = d^p - \frac{1}{2}p(p + 1) \) plus \( m = \dim \text{Pol}_\delta(\mathbb{R}^p) \). This sum can be kept low by choosing \( p \) small enough. Table 5 shows that the total dimension of Equation (8) remains moderate compared to the corresponding figures of the full polynomial basis from Table 4.

The calculation of the conditional expectation (5) is not as simple as for the full polynomial basis in Equation (16). Instead we need to compute the unconditional moments (11), which here reduce to

\[
G_{A,t,i}(x_1, \ldots, x_t) = \mathbb{E}^{Q}[g_i(\sum_{s=1}^t A_s^\top x_s + \sum_{s=t+1}^T A_s^\top X_s)],
\]

for \( A \in V_p(\mathbb{R}^d) \). Evaluation of Equation (18) boils down to compute multivariate moments of the \( \mathbb{R}^p \)-valued random variable \( Y = \sum_{s=1}^t A_s^\top x_s + \sum_{s=t+1}^T A_s^\top X_s \). Thereto, we utilize Kan (2008, Lemma 1), which relates multivariate moments to univariate moments, generalizing

\[
y^{\alpha} = y_1^{\alpha_1} \cdots y_p^{\alpha_p} = \frac{1}{|\alpha|!} \sum_{\nu = 0}^\alpha (-1)^{|\nu|}(\alpha_1)_{\nu_1} \cdots (\alpha_p)_{\nu_p}(h_{\alpha,\nu}^\top y)^{|\alpha|},
\]

for vectors \( h_{\alpha, \nu} = (\alpha_1/2 - \nu_1, \ldots, \alpha_p/2 - \nu_p)^\top \). The sum in (19) has in effect \( (\alpha_1 + 1) \cdots (\alpha_p + 1)/2 \) terms. For \( \delta = 3 \), this amounts to maximal \( 2^3/2 = 4 \) terms. As a result, the evaluation of (18), that is, the computation of \( \mathbb{E}^{Q}[Y^{\alpha}] \), reduces to the calculation of the \( |\alpha| \)th moments of the scalar random variables \( h_{\alpha,\nu}^\top Y \) in (19), which are given in closed form for various distributions of \( X \).

Example 4.6. For the multinormal case \( \text{vec}(X) \sim N(0, I_{dT}) \) we have

\[
h_{\alpha,\nu}^\top Y \sim N(\sum_{s=1}^t h_{\alpha,\nu}^\top A_s^\top X_s, \sum_{s=t+1}^T \|A_s h_{\alpha,\nu}\|^2).
\]

These univariate moments are given in closed form, as explicitly stated in Kan (2008, Proposition 2).
4.5 | Shallow neural network

In this third specification, we consider a shallow neural network with the rectified linear unit (ReLU) activation function. More specifically, we let \( p = m \) and \( g_i(y) = y_i^+ \), for \( i = 1, \ldots, m \). This yields the feature maps (9) of the form

\[
\phi_\theta(x) = (a_i^T x + b_i)^+,
\]

for weight matrices \( A = (a_1, \ldots, a_m) \in \mathbb{R}^{dT \times m} \) and bias vectors \( b \in \mathbb{R}^m \). Henceforth, we assume that Equation (13) holds for \( \delta = 1 \), so that \( \phi_\theta \in L_2^{\mathbb{Q}} \). By the positive homogeneity of the components of the feature map in the parameter, \( \phi_\theta(\lambda a_i, \lambda b_i) = \lambda \phi_\theta(a_i, b_i) \) for all \( \lambda > 0 \), we can assume that \( (a_i, b_i) \) lies in the unit sphere \( S_{dT} \) in \( \mathbb{R}^{dT+1} \). Hence the parameter set can be chosen as the compact product manifold \( \Theta = (S_{dT})^m \) without loss of generality.

What about linear independence of \( \phi_\theta(a_i, b_i) \)? Here is a fundamental result, which seems to be little known in the literature.

**Theorem 4.7.** For any \((a_i, b_i), (\tilde{a}_i, \tilde{b}_i) \in S_{dT}, i = 1, \ldots, m, \) the following statements hold:

(i) If

\[
\text{span}\{\phi_\theta(a_1, b_1), \ldots, \phi_\theta(a_m, b_m)\} = \text{span}\{\phi_\theta(\tilde{a}_1, \tilde{b}_1), \ldots, \phi_\theta(\tilde{a}_m, \tilde{b}_m)\},
\]

then \( \{\pm(a_1, b_1), \ldots, \pm(a_m, b_m)\} = \{\pm(\tilde{a}_1, \tilde{b}_1), \ldots, \pm(\tilde{a}_m, \tilde{b}_m)\} \).

(ii) If \((a_i, b_i) \neq \pm(a_j, b_j) \) for all \( i \neq j \) then

\[
\{\phi_\theta(a_1, b_1), \ldots, \phi_\theta(a_m, b_m)\} \text{ is linearly independent.}
\]

Note that the converse implication in Theorem 4.7(i) is not true, as can easily be seen from the case where \( m = 1 \) and \((\tilde{a}_1, \tilde{b}_1) = -(a_1, b_1) \in S_{dT} \) with \( a_1 \neq 0 \). What’s more, the following example shows that the assumptions in Theorem 4.7(ii) cannot be relaxed to pairwise inequality, \((a_i, b_i) \neq (a_j, b_j) \) for all \( i \neq j \).

**Example 4.8.** Let \((a_1, b_1), \ldots, (a_3, b_3) \in S_{dT} \) be linearly dependent vectors such that

\[
\sum_{i=1}^3 c_i(a_i, b_i) = 0 \text{ for some coefficients } c_i \neq 0.
\]

Define \((a_{3+i}, b_{3+i}) = -(a_i, b_i) \in S_{dT}, i = 1, 2, 3. \) Then \( \{\phi_\theta(a_i, b_i) \mid i = 1, \ldots, 6\} \) is linearly dependent,

\[
\sum_{i=1}^3 c_i \phi_\theta(a_i, b_i) + \sum_{i=1}^3 (-c_i) \phi_\theta(a_{3+i}, b_{3+i}) = \sum_{i=1}^3 c_i (a_i^T x + b_i) = 0,
\]

while \((a_i, b_i) \neq (a_j, b_j) \) for all \( i \neq j \).

We conclude that the assumptions of the existence Lemma 4.1 are not met. Indeed, we have the following non-existence result, which contrasts somewhat surprisingly with the widespread use of ReLU neural networks in machine learning.
Theorem 4.9. For the shallow ReLU neural network, there exists no minimizer of Equation (8) in general. Moreover, uniqueness does not hold, in the sense that the optimal subspace span{\(\phi(a_1, b_1), \ldots, \phi(a_m, b_m)\)} is not unique, in general.

Remark 4.10. The proof of the nonexistence statement in Theorem 4.9 is by means of a counterexample. It shows that the space of functions represented by the finite shallow neural network \(\{\phi^\top \beta \mid (\beta, \beta) \in \Theta \times \mathbb{R}^m\}\) is not closed in \(L^2_Q\) in general. This result also holds for finite neural networks with several layers, as recently shown in Petersen et al. (2021). Note that this finding is consistent with the celebrated universal approximation property of neural networks Hornik et al. (1989), which holds asymptotically for \(m \to \infty\). Indeed, in our case, \(m\) is fixed and finite. In view of their spectacular performance in solving practical problems, neural networks have become the subject of intensive research. We refer to the literature overview given in Petersen et al. (2021). An important strand of research is focused on the training of neural networks. Despite the nonconvexity of the objective function (8) in \((\beta, \beta)\), researchers have shown that its empirical landscape does not exhibit nonglobal local minima with high probability if the problem is sufficiently overparametrized. That is, if \(m\) is sufficiently large compared to the sample size, see Venturi et al. (2019). Also, we exploit these good empirical properties of neural networks in the numerical case studies below. Another line of research focuses directly on financial payoffs. Bossu et al. (2021) show that an infinitely large shallow neural network is capable of uniquely replicating any payoff function, and they find the representation in terms of the spectral decomposition of a Volterra integral equation.

The conditional expectation functions (11) read here as

\[
G_{(a_i, b_i), t}(x_1, \ldots, x_t) = \mathbb{E}_Q[(b_i + \sum_{s=1}^{t} a_{i,s}^\top x_s + \sum_{s=t+1}^{T} a_{i,s}^\top X_s)^+],
\]

where we decompose every column vector \(a_i\) of \(A\) into \(T\) consecutive blocks \(a_{i,t}\) such that \(a_i^\top = (a_{i,1}^\top, \ldots, a_{i,T}^\top)\). Evaluation of Equation (25) boils down to compute \(\mathbb{E}_Q[Y^+]\) for the scalar random variable \(Y = b_i + \sum_{s=1}^{t} a_{i,s}^\top x_s + \sum_{s=t+1}^{T} a_{i,s}^\top X_s\), which is given in closed form for various distributions of \(X\).

Example 4.11. For the multinormal case, \(\text{vec}(X) \sim N(0, I_{dT})\), we have

\[
Y \sim N(b_i + \sum_{s=1}^{t} a_{i,s}^\top x_s, \sum_{s=t+1}^{T} \|a_{i,s}\|^2).
\]

We then obtain a closed form expression for Equation (25) by combining Equation (26) with the well known Bachelier’s call option price formula \(\mathbb{E}_Q[Z^+] = \mu \Phi(\mu/\sigma) + \sigma \Phi'(\mu/\sigma)\), for a normal distributed random variable \(Z \sim N(\mu, \sigma^2)\), where \(\Phi\) denotes the standard normal distribution function and \(\Phi'\) its density function, see, for example, Delbaen and Schachermayer (2006, Section 4.3) or Fernandez-Arjona (2021).

In other cases, where the extended Fourier transforms \(\hat{Q}_t(u) = \mathbb{E}_Q[e^{u^\top X_t}]\) of the marginal distributions \(Q_t\) of \(X_t\) are given in closed form, for a suitable domain of complex-vector valued arguments \(u\), we can utilize Fourier transform analysis. Indeed, for any constant \(w > 0\), we have
the identity

\[ y^+ = \frac{1}{2\pi} \int_{\mathbb{R}} e^{(w + i\lambda)y} \frac{1}{(w + i\lambda)^2} d\lambda. \tag{27} \]

Hence the evaluation of Equation (25) reduces to the computation of the line integral

\[ G_{\{a_i, b_i\}, t}(x_1, \ldots, x_t) = \frac{1}{2\pi} \int_{\mathbb{R}} \hat{P}_Y(w + i\lambda) \frac{1}{(w + i\lambda)^2} d\lambda, \tag{28} \]

where \( \hat{P}_Y(w + i\lambda) = \mathbb{E}_Q[e^{(w + i\lambda)Y}] \) is in closed form. Note that Fourier type integrals like the one in Equation (28) are routinely computed in finance applications, for example, in Lévy type or affine models, Duffie et al. (2003). So one can draw on existing libraries of computer code.

### 4.6 Finite-sample estimation

While surprising and remarkable, the nonuniqueness and nonexistence results in Theorems 4.5 and 4.9 for polynomial feature maps with dimensionality reduction and shallow ReLU neural networks, respectively, are mainly of theoretical interest, arguably. See also Remark 4.10. In practice, we solve Equation (8) numerically using some quasi-Newton algorithm, which finds local minima that serve as approximate solutions. Thereto, we replace the model population measure \( Q \) by the empirical measure \( \hat{Q} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x^{(i)}} \) based on a training sample \( x^{(1)}, \ldots, x^{(n)} \) drawn from \( Q \), along with the corresponding function values \( y_i = f(x^{(i)}) \).

For the full polynomial basis, problem (8) boils down to the projection (4), and we obtain the optimal

\[ \hat{\beta} = (\frac{1}{n} \Phi^T \Phi)^{-1} (\frac{1}{n} \Phi^T y), \tag{29} \]

where we define \( \Phi \in \mathbb{R}^{n \times m} \) by \( \Phi_{ij} = \phi_j(x^{(i)}) \). This empirical estimator is consistent. The law of large numbers implies that \( \hat{\beta} \) converges in probability to the optimal \( \beta \) in Equation (4) for the model population measure \( Q \), as the sample size \( n \to \infty \). Moreover, the central limit theorem holds and theoretical guarantees for the sample error can be established, see, for example, Boudabsa and Filipović (2021).

For the polynomial feature map with dimensionality reduction, we use the Riemannian BFGS algorithm Huang et al. (2015) to find a local minimizer of Equation (8) over the Riemannian manifold \( V_p(\mathbb{R}^{dT}) \times \mathbb{R}^m \).

For the shallow ReLU neural network, we use the BFGS algorithm in the Scikit-learn library Pedregosa et al. (2011) for the Python programming language to find a local minimizer of Equation (8) over the full parameter set \( \mathbb{R}^{dT \times m} \times \mathbb{R}^m \times \mathbb{R}^m \).

Given the lack of uniqueness, and even existence, for the polynomial feature map with dimensionality reduction, and the shallow ReLU neural network, it remains an open research question whether asymptotic consistency holds and theoretical guarantees can be established for these specifications.
TABLE 6  European call, comparison of present value MApE (in percentage points)

| Samples | Full polynomial basis | LDR | Neural network |
|---------|-----------------------|-----|---------------|
|         | nMC Regress-now Regress-later | Regress-later | Regress-later |
| T = 5   |                       |     |               |
| 1000    | 6.5 4.0               | 1.3 | 0.5 4.2 0.2   |
| 5000    | 4.0 1.8               | 0.2 | 0.2 1.8 0.1   |
| 10,000  | 3.8 1.2               | 0.1 | 0.1 1.2 0.1   |
| 50,000  | 1.7 0.6               | 0.1 | 0.1 0.6 <0.1  |
| T = 40  |                       |     |               |
| 1000    | 7.5 7.2               | 3.8 | 7.9 5.5       |
| 5000    | 3.9 3.2               | 2.1 | 3.2 1.7       |
| 10,000  | 3.1 2.3               | 1.4 | 2.4 0.9       |
| 50,000  | 2.4 1.0               | 0.5 | 1.0 0.2       |

TABLE 7  European call, comparison of expected shortfall MApE (in percentage points)

| Samples | Full polynomial basis | LDR | Neural network |
|---------|-----------------------|-----|---------------|
|         | nMC Regress-now Regress-later | Regress-later | Regress-later |
| T = 5   |                       |     |               |
| 1000    | 26.7 21.5             | 4.8 | 2.0 47.5 0.9  |
| 5000    | 15.9 9.3              | 1.1 | 0.9 11.5 0.2  |
| 10,000  | 14.5 6.5              | 0.8 | 0.7 7.3 0.1   |
| 50,000  | 7.9 2.9               | 0.6 | 0.5 3.6 <0.1  |
| T = 40  |                       |     |               |
| 1000    | 403.6 141.0           | 10.0| 459.0 16.0    |
| 5000    | 110.9 46.8            | 6.5 | 106.9 4.9     |
| 10,000  | 56.6 29.6             | 4.9 | 57.3 3.3      |
| 50,000  | 19.1 12.5             | 2.9 | 11.4 1.0      |

5  | EUROPEAN CALL OPTION EXAMPLE REVISITED

Having presented the theoretical background, we now turn back to our illustrating preliminary example in Section 3. We apply the functional bases described in Sections 4.3–4.5, following the steps outlined in Section 4.6. For each functional basis we show the same quality metrics as in Section 3 in order to compare to the results from nMC estimation.

Additionally, each functional basis is also compared to other related methods, such as regress-now LSMC. When comparing between regression-based methods, we use an additional quality metric: the mean $L_1$ error over the empirical distribution of $\Delta V_1$. This metric allows us to make comparisons of the goodness-of-fit along the entire distribution, not only the tails. For more details about the quality metrics use, we refer to Appendix A.

5.1  | Results

The first comparison uses the full polynomial basis described in Section 4.3. In Tables 6 and 7, we present the MApE comparison among nMC, regress-now polynomial basis and the
replicating martingale (regress-later) full polynomial basis. We can see how the replicating martingale outperforms the other two methods in the estimation of the present value and the 99% expected shortfall. For a more comprehensive comparison, we look at the mean $L_1$ error in Table 8. We can see that Table 7 confirms the conclusions from Table 8, namely that the replicating martingale estimators outperform the regress-now estimators. In this regard, we verify what others in the literature have reported before for regress-later estimators.

Tables 6 and 7 do not show results for the full polynomial basis under the replicating martingale approach for $T = 40$. The reason for this is the combinatorial explosion in the number of basis functions as the dimensionality of the problem grows. As shown in Table 4, the number of basis functions for $d = 3, \delta = 3, T = 40$ is 302,621. The number of samples would have to be at least of that magnitude, yielding a problem that, while feasible for some algorithms, is not necessarily practical in the real world. Our focus is to describe a method that shows good quality even for high-dimensional problem with a manageable number of samples. It is important to note that the regress-now approach does not suffer from this problem and shows a quality improvement over the Monte Carlo approach.

The problems with high dimensional cases described in the previous paragraph motivate our use of LDR, as described in Section 4.4. Table 5 shows that the number of parameters to be estimated can be greatly reduced, from 816 to 29 for $T = 5$ and from 302,621 to 134 for $T = 40$. Whether a function $f$ can be well-approximated by a polynomial basis with LDR for a small $\delta$ and $p$ depends on the nature of the function. Asymptotically, any function in $L^2_{\mathbb{Q}}$ can be approximated with arbitrary precision.

The optimization problem in Equation (8) is solved over the product manifold $V_k(\mathbb{R}^{dT}) \times \mathbb{R}^m$ rather over $\mathbb{R}^{dT \times k} \times \mathbb{R}^m$, which is supported by Theorem 4.4. This reduces the effective dimensionality of the problem and simplifies the calculation of the conditional expectation of $\Phi(X)$ in Equation (6). For these examples, we have used the Riemannian BFGS algorithm from Huang et al. (2015) using the C++ library published by the authors. Additionally, we tested another two algorithms, Riemannian Trust Regions (Absil et al., 2007) as implemented by the Python library pymanopt (Townsend et al., 2016), and Grassmann Gauss–Newton (Hokanson & Constantine, 2018) as implemented by the authors in the publicly available Python library. In all cases, Riemannian BFGS achieved better results.

| Samples | Full polynomial basis | LDR | Neural network |
|---------|-----------------------|-----|----------------|
|         | Regress-now | Regress-later | Regress-later | Regress-now | Regress-later |
| $T = 5$ |            |               |               |             |               |
| 1000    | 15.5       | 4.6           | 1.1           | 36.6        | 0.5           |
| 5000    | 6.8        | 0.9           | 0.6           | 12.5        | 0.4           |
| 10,000  | 4.8        | 0.7           | 0.6           | 8.0         | 0.4           |
| 50,000  | 2.2        | 0.5           | 0.5           | 2.9         | 0.4           |
| $T = 40$ |           |               |               |             |               |
| 1000    | 26.5       | 4.6           | 68.8          | 7.4         |               |
| 5000    | 11.9       | 2.6           | 24.0          | 2.2         |               |
| 10,000  | 8.5        | 2.0           | 14.9          | 1.4         |               |
| 50,000  | 3.9        | 1.3           | 4.1           | 0.8         |               |
For this example, we have chosen $p = 3$. We performed a sensitivity analysis on this parameter and found that a larger value might lead to better results in some cases but not in all cases. The full results are included in Appendix E.1. In summary, $p$ is an important hyper-parameter but it is not special in any way, which means that any robust method for selecting hyper-parameters—cross-validation, pilot simulation, sensitivity analysis—can be used. In Appendix E.1, we also provide a sensitivity analysis to the starting point of the optimization. Since BFGS is quasi-Newton method, it is not guaranteed to find a global minimum in a general case. In the case of the polynomial basis with LDR, we find that the selection of the starting point makes a big difference in the final result.

Tables 6 and 7 show the results of applying LDR ($p = 3$) to the European call problem and we can see how this method performs in relation to the other alternatives. We can see that the polynomial LDR has lower error than nMC and both regress-now and regress-later polynomials. It also becomes clear that the LDR approach allows high dimensional problems where the regress-later approach on a full polynomial basis would fail due to producing a very large number of basis functions, which leads to a computational problem due to time or memory constraints.

We next describe the results obtained using a neural network model as defined in Section 4.5. For this example, we have chosen to work with $m = 101$ nodes, whereof one is a bias term only, say $A_{101} = 0$. The total number of parameters is thus $1600 + 101 = 1701$ for $T = 5$, and $12,100 + 101 = 12,201$ for $T = 40$. This choice was made via cross-validation.

The neural network was optimized via backpropagation using the BFGS algorithm from Python’s popular library scikit-learn (Pedregosa et al., 2011). The results in Tables 6 and 7 show an excellent quality of the neural network replicating martingale in this example, outperforming every other choice, except for the risk calculations with a very low number of samples (1000). Also here we can see that the replicating martingale method (regress-later) outperforms the regress-now variation, the same way that it did for the polynomials.

In Figure 2, we can compare the empirical distribution of the errors for each method. This figure makes it easy to qualitatively assess the differences between the different methods, for example: the high variance of nMC vis-a-vis the lower variance of replicating martingales,
or the higher accuracy of regress-later methods compared to regress-now methods. We can also see that, despite the nonlinear optimization with random starting points involved, the neural network replicating martingale does not have qualitatively higher variance than the polynomial equivalents.

It is interesting to consider the structure of the neural network and polynomial models, to understand what they have in common and what they do not. As seen in Equation (9), both methods use a linear map to reduce the dimensionality of the input before applying a nonlinear function. The polynomial model is based on global polynomials while the neural network can be seen as a data-driven piece-wise linear model. While usually piece-wise linear models require a grid to be defined a priori, neural networks adjust the bias term to place the grid where it is most needed according to the input data.

A comparison of the runtimes is given in Appendix D.

6 | INSURANCE LIABILITY MODEL EXAMPLE

Having shown the effectiveness of learning the replicating martingale in the case of a European call option, we present now a more complex example: a variable annuity guarantee. Unlike the previous example, this one features path dependent cash flows at multiple points in time and also a dependency on a stochastic mortality model, rather than only stochastic market variables. The model has been built using models commonly in use in the insurance industry. The policyholder population is fictitious. We first describe the model. Then we present the numerical results following the same structure as for the European call option example in Section 5.

6.1 | Model

The insurance product being simulated is an investment account with a return premium on death guarantee. Every policyholder has an investment account. At each time period, the policyholders pay a premium, which is used to buy assets. These assets are deposited in the fund, divided into the different assets according to a fixed asset allocation, which is the same as the initial one. The value of the fund is driven by the inflows from premiums and the market value changes, which are driven by the interest rate, equity, and real estate models. At each time step, a number of policyholders die—as determined by the stochastic life table—and the investment fund is paid out to the beneficiaries. If the investment fund were below the guaranteed amount, the company will additionally pay the difference between the fund value and the guaranteed amount. The guaranteed amount is the simple sum of all the premiums paid over the life of the policy. Over the course of the simulation, the premiums paid gradually increase the guaranteed amount for each policy.

All policies have the same maturity date. In the short-term, low dimensional example, the maturity is $T = 5$. In the long term, high dimensional example, the maturity is $T = 40$. At maturity, the higher of the fund value and the guaranteed amount is paid out to all survivors.

The investment portfolio holds four assets: a 10-year zero coupon bond, a 20-year zero coupon bond, an equity index and a real estate index. The bonds are annually replaced such that the time to maturity remains constant.
The model is described by the following equations, where all financial variables are nominal amounts, unless otherwise stated. The discounted cash flow at \( t = 1, \ldots, T \) is given by

\[
\zeta_t = \begin{cases} 
D_t \max(A_t, G_t)/C_t, & t < T \\
L_{T-1} \max(A_T, G_T)/C_T, & t = T
\end{cases}
\] (30)

where

- \( D_t \): total dead in period \((t-1, t]\)
- \( L_t \): total of policyholders alive at time \( t \)
- \( A_t \): value of assets at time \( t \) (per policy)
- \( G_t \): guaranteed value at time \( t \) (per policy)
- \( C_t \): value of the cash account at time \( t \)
- \( T \): maturity date of the policies

The value of assets at \( t = 0, \ldots, T \) is given by

\[
A_t = \sum_{i=1}^{4} U_i^t V_i^t = \sum_{i=1}^{4} U_{t-1}^i V_{t-1}^i + P_t,
\] (31)

where

\[
V_i^t = \begin{cases} 
B(t, t + 10), & i = 1, \\
B(t, t + 20), & i = 2, \\
S_t, & i = 3, \\
RE_t, & i = 4,
\end{cases}
\quad \text{and} \quad
V_{i-1}^t = \begin{cases} 
B(t, t + 9), & i = 1, \\
B(t, t + 19), & i = 2, \\
V_i^t, & i = 3, 4,
\end{cases}
\] (32)

denotes the unit price of asset \( i \) at time \( t \), where we use the notation \( V_{i-1}^t \) to express the rolling over of the constant-maturity bond investments for \( i = 1, 2, \) and

- \( U_i^t = (U_{t-1}^i V_{i-1}^t + M_t P_t)/V_i^t \): number of units of asset \( i \) held in period \((t, t + 1]\) (per policy), where \( U_{i-1}^1 := 0 \)
- \( B(t, s) \): value at time \( t \) of a bond maturing at time \( s \)
- \( S_t \): value of equity index at time \( t \)
- \( RE_t \): value of real estate at time \( t \)
- \( M_t \): asset allocation mix, henceforth fixed to \( M = (\frac{1}{3}, \frac{1}{3}, \frac{1}{5}, \frac{2}{15}) \)
- \( P_t \): premium paid at \( t \) for period \((t, t + 1]\) (per policy)

The policy variables are given by

\[
P_t = 100, \quad \text{and} \quad G_t = \begin{cases} 
0, & t = 0 \\
G_{t-1} + P_{t-1}, & t \geq 1
\end{cases}
\] (33)
As for the demographic variables, the total dead and total alive are given by

\[ D_t = \sum_x D_t^x, \quad \text{and} \quad L_t = \sum_x L_t^x, \]  

(34)

where

- \( D_t^x = L_t^x - D_t^x \): total dead of age \( x \) at \( t-1 \) in period \((t-1,t]\)
- \( L_t^x = L_t^{x-1} - D_t^x \): total alive of age \( x \) at time \( t \), with \( L_0^x = 1000 \) for all \( x \in (30,70) \)
- \( q_x(t) \): death rate for age \( x \) at \( t-1 \) in period \((t-1,t]\).

In total, the stochastic driver \( X \) has \( d = 5 \) components: two for the interest rate model, one for the equity model, one for the real estate model, and one for the stochastic mortality. The interest rate and equity models, for \( B \) and \( S \), are those described in Appendix B and used in previous examples. The real estate model, for \( RE \), is the same as the equity model from Appendix B, but uses an independent stochastic driver and a lower volatility than the equity model. The stochastic mortality follows the Lee–Carter model (Lee and Carter, 1992) to provide a trend and random fluctuations over time. More specifically, we model the death rate as

\[ q_x(t) = 1 - e^{-m_x(t)}, \quad m_x(t) = e^{a_x + b_x k(t)} \]

\[ k(t) = k(t-1) - 0.365 + \varepsilon_t, \quad \varepsilon_t = 0.621 X^{(lc)}_t \]

(35)

(36)

(37)

where

- \( m_x(t) \): force of mortality at time \( t \) for age \( x \)
- \( X^{(lc)}_t \): component of the stochastic driver \( X \) at time \( t \) used for mortality model
- \( a_x \) and \( b_x \): Lee–Carter parameters (table in Section B.3).

### 6.2 Results

The results for the variable annuity guarantee confirm those of the European call option example: the replicating martingale works very well, in particular the neural network model, which provides the best results in most cases. However, the more complex example also shows some limitations of the methods.

In the estimation of the present value, Table 9 shows that nMC is still very effective, but regression-based methods provide slightly better accuracy. The neural network model performs relatively badly in the case with the lowest number of samples (1000) and high dimensions (\( T = 40 \)), providing the worst results in that case. This is caused by over-fitting, as we describe in the analysis of the mean relative \( L_1 \) error below. Indeed, as for the European option example, by cross-validation, we have chosen \( m = 101 \) nodes, whereof one is a bias term only, say \( A_{101} = 0 \). The total number of parameters is thus 2600 + 101 = 2701 for \( T = 5 \), and 20, 100 + 101 = 20, 201 for \( T = 40 \). Alternative specifications of the width of the neural network are discussed in Appendix E.2. The quality reaches that of the other methods as the number of samples increase. Finally, we observe that the polynomial LDR method—which is calculated with \( p = 10 \)—shows its advantage over the full polynomial basis not only in being able to solve the high dimensional case, but
also in the estimation of the low dimensional case with low number of samples. The full polynomial basis has a MApE of 61% due to the basis containing 3276 elements, see Table 4, which exceeds the 1000 available samples. The polynomial LDR has a MApE of less than 0.1% due to only containing 286 basis elements.

In the estimation of the expected shortfall, shown in Table 10 and Figure 3, and the analysis of the mean relative $L_1$ error, shown in Table 11, we can observe that regress-later methods dominate over regress-now methods and nMC, with better mean absolute error and standard deviation. Unlike the case in the European call example where neural networks completely dominated the quality comparison, polynomial LDR shows better results in a few cases. However, which method shows better results are very sensitive to the choice of hyper-parameters. We provide a sensitivity analysis for hyper-parameters in Appendix E. Overall, neural networks have more room for improvement with an alternative choice of hyper-parameters and can be assumed to produce better results in this variable annuity example. We can observe several cases where an insufficient number of training samples leads to over-fitting and poor out-of-sample results. For example, for the full polynomial basis and $T = 5$, we find a large improvement in results when the training data changes from 1000 samples to 5000 samples. This basis has 3276 elements, see Table 4,
FIGURE 3  Distribution of expected shortfall estimates per method for the insurance liability example. Boxes show the upper and lower quartiles of the empirical distributions, while whiskers show their maxima and minima. Due to the logarithmic scale, a visual artifact is introduced by which the inter-quartile range seems to increase with larger sample sizes—most notably in the nested Monte Carlo (nMC) for Maturity = 40. In reality, the inter-quartile range decreases with larger sample sizes, but the logarithmic scale makes the box bigger for smaller errors [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 11  Insurance liability, comparison of relative mean $L_1$ error (in percentage points)

| Samples  | $T = 5$                  | $T = 40$                  |
|----------|--------------------------|--------------------------|
|          | Full polynomial basis    | LDR                      | Neural network          |
|          | Regress-now | Regress-later | Regress-later | Regress-now | Regress-later | Regress-later | Regress-later |
| 1000     | 1.6         | 61.0       | 0.2          | 4.7         | 0.1          |
| 5000     | 0.7         | <0.1       | 0.2          | 1.6         | 0.1          |
| 10,000   | 0.5         | <0.1       | 0.2          | 1.0         | <0.1         |
| 50,000   | 0.2         | <0.1       | 0.2          | 0.3         | <0.1         |
| 1000     | 3.3         | 0.8        | 10.0         | 6.1         |
| 5000     | 1.4         | 0.3        | 3.4          | 0.6         |
| 10,000   | 1.0         | 0.4        | 2.2          | 0.5         |
| 50,000   | 0.4         | 0.4        | 0.6          | 0.1         |

which means that when working with 1000 samples, we have more parameters than samples. The same effect can be seen in the neural network replicating martingale for $T = 40$ when the sample size changes from 10,000 to 50,000 samples. This can be explained by the fact that this model has 20,201 parameters, as mentioned above. In some cases, for example, the case neural network regress-later estimator for $T = 5$, the MApE ES increases when the sample size increases from 5000 to 10,000 and 50,000, see Table 10. This behavior is not present in the relative mean $L_1$ error, as evidenced in Table 11. This is due to the divergence between the error being minimized—errors along the full cash flows distribution—and the error being measured—errors in the tail of $T = 1$ conditional expectation distribution.

A comparison of the runtimes for the insurance example is important to determine the relative strength of the methods as feasible solution in the real world. Details are given in Appendix D.
7 | CONCLUSION

In the context of the need for accurate and fast calculations in portfolio pricing and risk management, we have introduced a data-driven method to build replicating martingales under different functional basis. This method yields lower errors than standard nMC simulation.

The model learns the features necessary for an effective low-dimensional representation from finite samples in a supervised setting. By doing so, it can be very effective in high-dimensional problems, without some of the usual difficulties associated with them.

We have presented two examples to demonstrate the usefulness of replicating martingales in the calculation of economic capital. The first is a typical benchmark example for calculations involving financial derivatives: a European call option. The second is a path-dependent insurance product, a variable annuity guarantee. Replicating martingales outperform other methods in the literature and in use in the financial industry for these two representative cases. This is illustrated by extensive comparisons and sensitivity analyses.

ACKNOWLEDGMENTS

We thank participants at the Online Workshop on Stochastic Analysis and Hermite Sobolev Spaces, SFI Research Days, SIAM Conference on Financial Mathematics and Engineering, Virtual Risk Management and Insurance Seminar at Georgia State University, SIAM Activity Group on Financial Mathematics and Engineering virtual seminar, Ben Feng, Antoon Pelsser, and two anonymous referees for their comments.

Open access funding provided by Ecole Polytechnique Federale de Lausanne.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are openly available in Fernandez-Arjona and Filipović (2022).

NOTES

1There is a closed-form solution for this particular example but our intention is to exemplify the general case, which does not have one.

2In keeping with industry convention, the present value is reported as positive number, $|V_0|$, without taking into account the fact that it is a short position.

3Strictly speaking, the dimension of the linear span of the functions $\phi_i$ in $L_2^Q$ could be less than $m$, because they may be linearly dependent as elements in $L_2^Q$. This is in particular the case when $Q$ is an empirical measure from a sample of size $n$, as described in Section 4.6. In this case, $m > n$ would simply lead to an exact interpolation of $f$, which likely will result in overfitting.

4This index mapping only performs the ordering of the elements, which is required to conveniently and formally write the conditional expectation function. This index mapping does not perform any selection of a subset of the basis and any mapping would yield the same final results.

5We also complete some arguments in He et al. (2020), who do not explain what kind of derivative “$\nabla \sum_{i=1}^m c_i \phi(a_i, b_i)$” stands for.

REFERENCES

Absil, P. A., Baker, C. G., & Gallivan, K. A. (2007). Trust-region methods on Riemannian manifolds. Foundations of Computational Mathematics, 7(3), 303–330.

Andreatta, G., & Corradin, S. (2003). Valuing the surrender options embedded in a portfolio of italian life guaranteed participating policies: A least squares Monte Carlo approach. In Proceedings of the “Real option theory meets practice”, 8th annual international conference, Montreal.
Bosu, S., Carr, P., & Papanicolaou, A. (2021). A functional analysis approach to the static replication of European options. *Quantitative Finance, 21*(4), 637–655.

Boudabsa, L., & Filipović, D. (2021). Machine learning with kernels for portfolio valuation and risk management. *Finance and Stochastics, 26*, 131–172.

Broadie, M., Du, Y., & Moalem, C. C. (2015). Risk estimation via regression. *Operations Research, 63*(5), 1077–1097.

Carriere, J. F. (1996). Valuation of the early-exercise price for options using simulations and nonparametric regression. *Insurance: Mathematics and Economics, 19*(1), 19–30.

Castellani, G., Fiore, U., Marino, Z., Passalacqua, L., Perla, F., Scognamiglio, S., & Zanetti, P. (2018). An investigation of machine learning approaches in the solvency II valuation framework. Available at SSRN 3303296.

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)?—Arguments against avoiding RMSE in the literature. *Geoscientific Model Development, 7*(3), 1247–1250.

Cunningham, J. P., & Ghahramani, Z. (2015). Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research, 16*(1), 2859–2900.

Delbaen, F., & Schachermayer, W. (2006). *The mathematics of arbitrage*. Springer finance. Springer-Verlag.

Duffie, D., Filipović, D., & Schachermayer, W. (2003). Affine processes and applications in finance. *The Annals of Applied Probability, 13*(3), 984–1053.

Duong, Q. D. (2019). Application of Bayesian penalized spline regression for internal modeling in life insurance. *European Actuarial Journal, 9*(1), 67–107.

Fernandez-Arjona, L. (2021). A neural network model for solvency calculations in life insurance. *Annals of Actuarial Science, 15*(2), 259–275. https://doi.org/10.1017/S1748499520000330

Fernandez-Arjona, L., & Filipović, D. (2022). Reproducibility data for tables in “A machine learning approach to portfolio pricing and risk management for high dimensional problems”, [Dataset].

Föllmer, H., & Schied, A. (2004). *Stochastic finance. An introduction in discrete time (extended ed.), volume 27 of De Gruyter Studies in Mathematics*. Walter de Gruyter & Co.

Glasserman, P. (2013). *Monte Carlo methods in financial engineering* (Vol. 53). Springer Science & Business Media.

Glasserman, P., & Yu, B. (2002). Simulation for American options: Regression now or regression later? In *Monte Carlo and quasi-Monte Carlo Methods 2002* (pp. 213–226). Springer.

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

Ha, H., & Bauer, D. (2020). *A least-squares Monte Carlo approach to the estimation of enterprise risk*. https://danielbaueracademic.files.wordpress.com/2020/04/lsm_habauer2020.pdf

Hanin, B., & Sellke, M. (2017). Approximating continuous functions by ReLU nets of minimal width. *arXiv preprint arXiv:1710.11278*

He, J., Li, L., Xu, J., & Zheng, C. (2020). ReLU deep neural networks and linear finite elements. *Journal of Computational Mathematics, 38*(3), 502–527.

Hokanson, J. M., & Constantine, P. G. (2018). Data-driven polynomial ridge approximation using variable projection. *SIAM Journal on Scientific Computing, 40*(3), A1566–A1589.

Hong, L. J., Juneja, S., & Liu, G. (2017). Kernel smoothing for nested estimation with application to portfolio risk measurement. *Operations Research, 65*(3), 657–673.

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks, 2*(5), 359–366.

Huang, W., Gallivan, K. A., & Absil, P. A. (2015). A Broyden class of quasi-Newton methods for Riemannian optimization. *SIAM Journal on Optimization, 25*(3), 1660–1685.

Kan, R. (2008). From moments of sum to moments of product. *Journal of Multivariate Analysis, 99*(3), 542–554.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278–2324.

Lee, R. D., & Carter, L. R. (1992). Modeling and forecasting U.S. mortality. *Journal of the American Statistical Association, 87*(419), 659–671.

Lee, S. H., & Glynn, P. W. (2003). Computing the distribution function of a conditional expectation via Monte Carlo: Discrete conditioning spaces. *ACM Transactions on Modeling and Computer Simulation (TOMACS), 13*(3), 238–258.

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

Since the focus on this paper are applications in pricing and risk managements, we use two key quality metrics. The first one looks into the goodness of fit in the tail of the distribution, and the second one, the goodness of fit across the body of the distribution. For the tail of the distribution, we look at expected shortfall and value at risk, for the loss-making tail. For the body of the distribution, we look at the $L_1$ error.

We treat the models as statistical estimators since their estimates are subject to the randomness of their inputs. For that reason, for each of those metrics described above, we derive an empirical distribution based on $R$ macro-runs of the entire simulation-estimation-prediction chain of calculations. That means that we also need to define which metric summarizes the results of the empirical distribution. In both cases (tail error and $L_1$ error), we use the mean absolute error. In all cases, we work with relative errors, expressed as a percentage. Root mean squared errors would have been an option but the advantages of the mean absolute error have been well-documented in Willmott and Matsuura (2005) and Chai and Draxler (2014).

In the sections below, we describe in detail the calculation of our two quality metrics: mean absolute percentage error on tail error (MApE), and mean relative $L_1$ error.

A.1 Mean absolute percentage error

Let us consider an empirical distribution of $X_{1:1}$, composed of $n$ samples. From this distribution, we can obtain an empirical distribution of $V_j$. Given a function $f^*$, we obtain $R$ repetitions of its finite sample estimator $\hat{f}$. For each function in these $\{f_j^R\}_{j=1}^R$, we can produce an empirical
distribution of its value estimator $\hat{V}_t = \mathbb{E}^Q_t [\hat{f}(X)]$ using $X_{1:t}$, therefore, obtaining a set of empirical distributions $\{\hat{V}^{(j)}_t\}_{j=1}^R$.

Given a benchmark expected shortfall calculation at $\alpha$ (e.g., $\alpha = 99\%$) confidence, $\text{ES}_\alpha[-\Delta V_I]$, an estimator of such quantity $\text{ES}_\alpha[-\Delta \hat{V}^{(j)}_t]$, and $R$ repetitions (independent samples) of such estimator $j = 1 \ldots R$, the mean absolute percentage error (MApE) is defined as

$$\text{MApE ES} = \frac{1}{R} \sum_{j=1}^R \frac{\left| \text{ES}_\alpha[-\Delta \hat{V}^{(j)}_t] - \text{ES}_\alpha[-\Delta V_I] \right|}{\text{ES}_\alpha[-\Delta V_I]}.$$ (A.1)

The MApE metric can also be applied to the present value of $V_t$, $\mathbb{E}[V_t] = V_0$:

$$\text{MApE PV} = \frac{1}{R} \sum_{j=1}^R \frac{|\hat{V}^{(j)}_0 - V_0|}{V_0}.$$ (A.2)

### A.2 | Mean relative $L_1$ error

Given the above, the mean relative $L_1$ error is defined as

$$\frac{1}{R} \sum_{j=1}^R \frac{\mathbb{E}[|\hat{V}^{(j)}_t - V_t|]}{\mathbb{E}[|V_t|]}.$$ (A.3)

This metric is related to the error on the expected shortfall in the following way:

$$|\text{ES}_\alpha[\hat{V}_t] - \text{ES}_\alpha[V_t]| \leq \frac{1}{\alpha} \mathbb{E}[|\hat{V}_t - V_t|].$$ (A.4)

This shows that for any $\alpha$, the expected shortfall MApE is bounded by a multiple of the $L_1$ error. While the MApE ES is a metric calculated for a particular $\alpha$ and only takes into account the distribution beyond the $\alpha$th percentile, the mean relative $L_1$ error takes into account the whole distribution and bounds the expected shortfall error for any $\alpha$.

### APPENDIX B: ECONOMIC SCENARIO GENERATOR

We describe the basic financial models underlying the economic scenario generator of the examples in this paper. We assume that the stochastic driver is multinormal $\text{vec}(X) \sim N(0, I_d T)$, and consider either $d = 3$ or $d = 5$.

#### B.1 | Interest rate model

The interest rate model is based on the continuous time Hull–White short rate model

$$dr_t = \kappa(b(t) - r_t)dt + \sigma dW_t,$$ (B.1)

for parameters $\kappa$, $\sigma$, and function $b(t)$, where $W$ denotes a Brownian motion under the risk-neutral measure, see, for example, Glasserman (2013). The nominal price at time $t$ of a zero-coupon bond with maturity $T$ is given by

$$B(t, T) = \exp(-A(t, T)r_t + C(t, T)),$$ (B.2)
where

\[
A(t, T) = \frac{1}{\kappa}'(1 - e^{-\kappa(T-t)}),
\]

and where we denote \( h(t, T) = \int_t^T e^{-\kappa(u-t)} b(s) ds u. \)

In discrete time, we exactly simulate the short rate, \( r_t \), and log-cash account, \( Y_t = \int_0^t r_u du \), jointly from the above Hull–White model according to the formulas in Glasserman (2013), which are based on a two-dimensional Gaussian stochastic driver. We, therefore, use the first two components of \( X \), that is, \( X_{1,t} \) and \( X_{2,t} \), as follows. As for \( r_t \), define

\[
g(t) = \int_t^{t+1} e^{-\kappa(t+1-s)} b(s) ds,
\]

\[
\sigma_r = \frac{\sigma^2}{2\kappa}'(1 - e^{-2\kappa}),
\]

and set \( r_{t+1} = e^{-\kappa} r_t + \kappa g(t) + \sigma_r X_{1,t+1}. \)

As for \( Y_t \), define

\[
h(t) = h(t, t + 1),
\]

\[
\sigma_Y = \frac{\sigma^2}{\kappa^2}'(1 + \frac{1}{2\kappa}'(1 - e^{-2\kappa}) + \frac{2}{\kappa}'(e^{-\kappa})),
\]

\[
\sigma_{rY} = \frac{\sigma^2}{2\kappa}'(1 + e^{-2\kappa} - 2e^{-\kappa}),
\]

\[
\rho_{rY} = \sigma_{rY} / (\sigma_r \sigma_Y),
\]

and set \( Y_{t+1} = Y_t + (1/\kappa)(1 - e^{-\kappa}) r_t + \kappa h(t) + \sigma_Y X'_{2,t+1} \), for the correlated driver \( X'_{2,t+1} = \rho_{rY} X_{1,t+1} + \sqrt{1 - \rho_{rY}^2} X_{2,t+1}. \)

**B.2 Equity and real estate index models**

For a given matrix \( \Sigma \) that encodes the desired correlations, we denote the correlated Gaussian stochastic driver \( X' = \Sigma X \). For the examples in this paper, we set \( X'_{1,t} = X_{1,t} \), \( X'_{2,t} \) as above, \( X'_{3,t} \), \( X'_{4,t} \) to be correlated with \( X_{1,t} \), and \( X'_{5,t} = X_t^{(lc)} = X_{5,t} \)—used for the mortality model—to be independent of all other variables.

For both, equity and real estate, a geometric Brownian process models the respective index excess return, with the recursive formula

\[
Z_{j,t} = Z_{j,t-1} \exp \left(-\frac{\sigma^2_j}{2} + \sigma_j X'_{j,t} \right),
\]
### Table B.1 Lee–Carter Parameters $a_x$ and $b_x$ for Every Age $x$

| $x$     | $a_x$       | $b_x$    |
|---------|-------------|----------|
| 0       | -3.641090   | 0.90640  |
| 1, 2, 3, 4 | -6.705810   | 0.11049  |
| 5, 6, 7, 8, 9 | -7.510640   | 0.09179  |
| 10, 11, 12, 13, 14 | -7.557170   | 0.08358  |
| 15, 16, 17, 18, 19 | -6.443340   | 0.05351  |
| 20, 21, 22, 23, 24 | -6.229090   | 0.04744  |
| 25, 26, 27, 28, 29 | -5.913250   | 0.05966  |
| 30, 31, 32, 33, 34 | -5.513230   | 0.04458  |
| 35, 36, 37, 38, 39 | -4.656800   | 0.06173  |
| 40, 41, 42, 43, 44 | -4.254970   | 0.03382  |
| 45, 46, 47, 48, 49 | -2.856080   | 0.02949  |
| 50, 51, 52, 53, 54 | -3.473130   | 0.02880  |
| 55, 56, 57, 58, 59 | -2.60170    | 0.02908  |
| 60, 61, 62, 63, 64 | -1.799600   | 0.03240  |
| 65, 66, 67, 68, 69 | -1.409363   | 0.03091  |
| 70, 71, 72, 73, 74 | -1.036550   | 0.03091  |
| 80, 81, 82, 83, 84 | -0.680350   | 0.03091  |
| 85, 86, 87, 88, 89 | -0.341050   | 0.03091  |

where $j = 3$ for the equity index and $j = 4$ for the real estate index. The equity index $S_t$ and the real estate index $RE_t$ are then given by $S_t = C_t Z_{3,t}$ and $RE_t = C_t Z_{4,t}$, respectively, where $C_t = \exp(Y_t)$ denotes the cash account.

#### B.3 Lee–Carter Parameters

The Lee–Carter parameters for the mortality model are based on the findings in the original paper Lee and Carter (1992), and they are shown in Table B.1.

#### APPENDIX C: PROOFS

This section contains all proofs and some auxiliary results of independent interest.

### C.1 Proof of Lemma 4.1

In view of Equation (4) and by orthogonality, we have $\|f - \phi_\theta^T \beta\|_{L_Q^2}^2 \geq \|f - \phi_\theta^T \beta_0\|_{L_Q^2}^2 = \|f\|_{L_Q^2}^2 - \langle f, \phi_\theta^T \beta_0 \rangle_{L_Q^2}$, for all $(\theta, \beta) \in \Theta \times \mathbb{R}^m$. On the other hand, by assumption (iii) we can write $\beta_0 = \langle \phi_\theta, \phi_\theta^T \rangle_{L_Q^2}^{-1} \langle \phi_\theta, f \rangle_{L_Q^2}$ and hence $\langle f, \phi_\theta^T \beta_0 \rangle_{L_Q^2} = \langle f, \phi_\theta^T \rangle_{L_Q^2} \langle \phi_\theta, \phi_\theta^T \rangle_{L_Q^2}^{-1} \langle \phi_\theta, f \rangle_{L_Q^2}$. Hence Equation (8)
is equivalent to Equation (4) and
\[
\max_{\theta \in \Theta} \langle f, \phi^T \rangle_{L^2_Q} \langle \phi_\theta, \phi^T \rangle_{L^2_Q}^{-1} \langle \phi_\theta, f \rangle_{L^2_Q}.
\] (C.1)

By the assumptions of the lemma, \( \theta \mapsto \langle f, \phi^T \rangle_{L^2_Q} \langle \phi_\theta, \phi^T \rangle_{L^2_Q}^{-1} \langle \phi_\theta, f \rangle_{L^2_Q} \) is continuous, and hence attains its maximum on the compact set \( \Theta \). This completes the proof.

C.2 Proof of Theorem 4.4

Theorem 4.4 follows from Lemmas C.1 and C.2 below.

**Lemma C.1.** Let \( A, \bar{A} \in \mathbb{R}^{dT \times p} \) with full rank and \( b, \bar{b} \in \mathbb{R}^p \). The following are equivalent:

(i) \( \ker A^T = \ker \bar{A}^T \)

(ii) \( \bar{A} = A S \), for some invertible \( p \times p \)-matrix \( S \)

(iii) \( \text{span}\{\phi_{(A,b),1}, \ldots, \phi_{(A,b),m}\} = \text{span}\{\phi_{(\bar{A},\bar{b}),1}, \ldots, \phi_{(\bar{A},\bar{b}),m}\} \)

**Proof.** (i)\(\iff\)(ii): is elementary.

(ii)\(\Rightarrow\)(iii): define \( p_i(y) = g_i(y + b) \) and \( \bar{p}_i(y) = g_i(S^T y + \bar{b}) \) and note that \( \{p_1, \ldots, p_m\} \) as well as \( \{\bar{p}_1, \ldots, \bar{p}_m\} \) forms a basis of \( \text{Pol}_\delta(\mathbb{R}^p) \). On the other hand, we have \( \phi_{(A,b),i}(x) = g_i(A^T x + b) = p_i(A^T x) \) and \( \phi_{(\bar{A},\bar{b}),i}(x) = g_i(S^T A^T x + \bar{b}) \). This yields the claim.

(iii)\(\Rightarrow\)(i): we argue by contradiction and assume that there exists some \( \bar{x} \in \ker \bar{A}^T \setminus \ker A^T \). Then \( y \mapsto y^T A^T \bar{x} \) is in \( \text{Pol}_\delta(\mathbb{R}^p) \) and can thus be written as linear combination \( y^T A^T \bar{x} = \sum_{i=1}^m c_i g_i(y + b) \). We obtain that the function \( h(x) = x^T A A^T \bar{x} = \sum_{i=1}^m c_i \phi_{(A,b),i}(x) \) lies in the span \( \phi_{(A,b)} \). But \( h \not\in \text{span} \phi_{(A,b)} \) because \( \phi_{(A,b),i}^T t \bar{x} = g_i(\bar{b}) \) is constant in \( t \in \mathbb{R} \), for all \( i \), while \( h(t \bar{x}) = \|A^T \bar{x}\|^2 t \) is not. This completes the proof.

**Lemma C.2.** Let \( A, \bar{A} \in \mathbb{R}^{dT \times p} \) with full rank. The following are equivalent:

(i) \( A \in V_p^0(\mathbb{R}^{dT}) \) and \( \bar{A} = A S \), for some invertible \( p \times p \)-matrix \( S \)

(ii) \( \bar{A} = A (A^T A)^{-1/2} U \) for some orthogonal \( p \times p \)-matrix \( U \)

**Proof.** (i)\(\Rightarrow\)(ii): we obtain \( I_p = \bar{A}^T \bar{A} = S^T A^T AS = S^T (A^T A)^{1/2} (A^T A)^{1/2} S \). Hence the \( p \times p \)-matrix \( U = (A^T A)^{1/2} S \) is orthogonal and \( S = (A^T A)^{-1/2} U \), which yields the claim.

(ii)\(\Rightarrow\)(i): is elementary.

C.3 Proof of Theorem 4.5

As \( V_p^0(\mathbb{R}^{dT}) \) is a compact manifold, it follows by inspection that the assumptions of Lemma 4.1 are met. Hence, there exists a minimizer of Equation (8). The nonuniqueness statement is proved by means of the following counterexample. Assume \( f(x) = f(V x) \) and the pushforward \( V^* Q = Q \) for some orthogonal \( dT \times dT \)-matrix \( V \). Then, for any \( A \in V_p^0(\mathbb{R}^{dT}) \) and \( \beta \in \mathbb{R}^m \), we have \( \|f - \phi_A^T \beta\|_{L^2_Q} = \|f - \phi_{V^T A}^T \beta\|_{L^2_Q} \). But \( \text{span}\{\phi_{A,1}, \ldots, \phi_{A,m}\} \neq \text{span}\{\phi_{V^T A,1}, \ldots, \phi_{V^T A,m}\} \) in general by Theorem 4.4. This completes the proof of Theorem 4.5.

C.4 Proof of Theorem 4.7

We follow the heuristic arguments of He et al. (2020). First, note that any linear combination \( h = \sum_{j=1}^m c_j \phi_{(a,j),b,j} \) is a continuous, piece-wise affine function. As such it is Bouligand differentiable on \( \mathbb{R}^{dT} \), see Scholtes (2012, Theorem 3.1.2). That is, its directional derivative \( \nabla_v h(x) = \lim_{\epsilon \to 0} (h(x + \)
\( \epsilon v - h(x)/\epsilon \) exists for all \( x, v \in \mathbb{R}^{dT} \), and it provides a first-order approximation, \( \lim_{y \to x} \| h(y) - h(x) - \nabla_y h(x) \| / \| y - x \| = 0 \). Accordingly, the classical calculus rules carry over and we have \( \nabla_v \sum_{j=1}^m c_j \phi(a_j, b_j) (x) = \sum_{j=1}^m c_j \nabla_v \phi(a_j, b_j) (x) \), see Scholtes (2012, Corollary 3.1.1).

Next, for a function \( h : \mathbb{R}^{dT} \to \mathbb{R} \), we denote by \( D_h \) the set of points of discontinuity. For \( h(x) = \nabla_v \phi(a_i, b_i) (x) \) and any \( v \in \mathbb{R}^{dT} \), we obtain

\[
D_{\nabla_v \phi(a_i, b_i)} = \begin{cases} \emptyset, & \text{if } v^\top a_i = 0, \\ H(a_i, b_i), & \text{otherwise}, \end{cases}
\]  

(C.2)

for the affine hyperplane \( H(a_i, b_i) = \{ x \mid a_i^\top x + b_i = 0 \} \).

Now assume \( \phi(a_i, b_i) \in \text{span}\{\phi(\tilde{a}_1, \tilde{b}_1), \ldots, \phi(\tilde{a}_m, \tilde{b}_m)\} \), so that \( \phi(a_i, b_i) = \sum_{j=1}^m c_j \phi(\tilde{a}_j, \tilde{b}_j) \) for some real coefficients \( c_j \). Then

\[
H(a_i, b_i) = D_{\nabla_v \phi(a_i, b_i)} = \bigcup_{j=1}^m D_{\nabla_v \phi(\tilde{a}_j, \tilde{b}_j)} = \bigcup_{j=1}^m H(\tilde{a}_j, \tilde{b}_j),
\]  

(C.3)

where we used the obvious relation \( D_{1_t h_1 + 2_t h_2} \subseteq D_{h_1} \cup D_{h_2} \) for functions \( h_1, h_2 \) and real coefficients \( t_1, t_2 \). This implies that \( (a_i, b_i) \in \{ \pm(a_1, b_1), \ldots, \pm(a_m, b_m) \} \). Since \( i \) was arbitrary, we obtain \( \{ \pm(a_1, b_1), \ldots, \pm(a_m, b_m) \} \subseteq \{ \pm(a_1, b_1), \ldots, \pm(a_m, b_m) \} \). A similar argument for \( (a_i, b_i) \) in lieu of \( (a_i, b_i) \) implies the converse inclusion. This proves (i).

For the proof of (ii), we argue by contradiction. Suppose Equation (23) does not hold, so that \( \phi(a_i, b_i) \notin \text{span}\{\phi(\tilde{a}_j, \tilde{b}_j) \mid j \neq i\} \) for some \( i \). Hence, Equation (22) holds for \( (a_j, b_j) : = (a, b) \) for all \( j 
eq i \), and \( (\tilde{a}_i, \tilde{b}_i) : = (a, b) \) for some \( j \neq i \). But then part (i) implies that \( (a_i, b_i) = \pm(a_j, b_j) \) for some \( j \neq i \), which contradicts the assumption of (ii). This completes the proof of Theorem 4.7.

**Remark C.3.** One may reckon that Equations (22) and (23) together imply

\[
\{(a_1, b_1), \ldots, (a_m, b_m)\} = \{(\tilde{a}_1, \tilde{b}_1), \ldots, (\tilde{a}_m, \tilde{b}_m)\}.
\]  

(C.4)

However, this is not true in general. Indeed, let \( (a_1, b_1), \ldots, (a_3, b_3) \in S_{dT} \) as in Example 4.8, and define \( S = \{(a_1, b_1), \pm(a_2, b_2), \pm(a_3, b_3)\} \) and \( \tilde{S} = \{-(a_1, b_1), \pm(a_2, b_2), \pm(a_3, b_3)\} \). Then \( F = \{\phi(a, b) \mid (a, b) \in S\} \) and \( \bar{F} = \{\phi(a, b) \mid (a, b) \in \tilde{S}\} \) are both linearly independent sets. On the other hand, we have \( c_1 \phi(a_1, b_1) = c_1 \phi(a_1, b_1) + \sum_{i=2}^3 c_i (\phi(a_i, b_i) - \phi(-a_i, b_i)) \), and hence \( \text{span} \ F = \text{span} \ F \bar{F} \). But \( S \neq \tilde{S} \).

**C.5 | Proof of Theorem 4.9**

We prove the theorem by means of two counterexamples. First, let \( d = T = 1, \Omega = N(0,1) \), and \( f(x) = 1_{[0,\infty)}(x) \). For the feature map, we let \( m = 2, \) and set \( b_n = 1/n, a_n = \sqrt{1 - b_n^2}, \) and \( \theta_n = ((a_n, b_n), (0,0)) \in (S_1)^2 \). Then \( \theta_n \to ((0,1), (0,1)) \) as \( n \to \infty \). On the other hand, for \( \beta_n = (1/b_n, -a_n/b_n)^T \), we have

\[
\phi_{\theta_n} (x)^T \beta_n = \frac{1}{b_n} (a_n x + b_n)^+ - \frac{a_n}{b_n} x^+ \to 1_{[0,\infty)}(x) \quad \text{in } L^2_\Omega \quad \text{as } n \to \infty.
\]  

(C.5)

Hence \( \inf_{(\theta, \beta) \in (S_1)^2 \times \mathbb{R}^2} \| f - \phi_{\theta}^T \beta \|_{L^2_\Omega} = \lim_n \| f - \phi_{\theta_n}^T \beta_n \|_{L^2_\Omega} = 0 \), but the infimum is not attained, \( \| f - \phi_{\theta}^T \beta \|_{L^2_\Omega} > 0 \) for all \( (\theta, \beta) \in (S_1)^2 \times \mathbb{R}^2 \). This proves the nonexistence statement.
For the nonuniqueness, assume \( f(x) = f(Vx) \) and the pushforward \( V_* \mathbb{Q} = \mathbb{Q} \) for some \( dT \times dT \)-matrix \( V \). Then, for any \( (A, b) \in (S_{dT})^m \) and \( \beta \in \mathbb{R}^m \), we have \( \| f - \phi^T(V^TA, b)\beta \|^2_{L^2_{\mathbb{Q}}} = \| f - \phi^T(A, b)\beta \|^2_{L^2_{\mathbb{Q}}} \). But \( \text{span}\{\phi(A, b), 1, \ldots, \phi(A, b), m\} \neq \text{span}\{\phi(V^TA, b), 1, \ldots, \phi(V^TA, b), m\} \) in general by Theorem 4.7. This completes the proof of Theorem 4.9.

APPENDIX D: COMPARISON OF RUNTIMES

We discuss how long it takes to run the training and prediction phases on each model from Sections 5 and 6. This involves: running the regression on the number of samples indicated on the first column and calculating \( \hat{V}_1(x_i) \) for 1000,000 validation samples.

Table D.1 shows the runtimes for the European call option example from Section 5. They clearly show the effect of the dimensionality reduction in the computational cost of the replicating martingale method.

Table D.2 shows the runtimes for the insurance liability example from Section 6. As above, unsurprisingly, we find regress-now methods to be faster than regress-later methods. This might partially explain the popularity with practitioners, especially for frequent calculations that do not require high precision. However, for quarterly or annual calculations of regulatory solvency,

**Table D.1** European call, comparison of runtime (in seconds), single core AMD Opteron 6380

| Samples | Full polynomial basis | LDR | Neural network |
|---------|-----------------------|-----|---------------|
|         | Regress-now | Regress-later | Regress-later | Regress-now | Regress-later |
| \( T = 5 \) | | | |
| 1000 | 0.9 | 1.5 | 1.4 | 9.5 | 18.4 |
| 5000 | 0.9 | 2.3 | 1.6 | 16.1 | 21.1 |
| 10,000 | 0.9 | 3.0 | 2.1 | 24.0 | 24.4 |
| 50,000 | 1.0 | 13.4 | 7.1 | 92.6 | 49.4 |
| \( T = 40 \) | | | |
| 1000 | 0.9 | 7.2 | 10.0 | 18.5 |
| 5000 | 0.9 | 22.3 | 15.5 | 24.4 |
| 10,000 | 1.0 | 51.7 | 24.0 | 30.0 |
| 50,000 | 1.4 | 271.7 | 19.1 | 76.1 |

**Table D.2** Insurance liability, comparison of runtime (in seconds), single core AMD Opteron 6380

| Samples | Full polynomial basis | LDR | Neural network |
|---------|-----------------------|-----|---------------|
|         | Regress-now | Regress-later | Regress-later | Regress-now | Regress-later |
| \( T = 5 \) | | | |
| 1000 | 0.8 | 4.3 | 30.4 | 4.0 | 5.5 |
| 5000 | 0.9 | 54.4 | 113.2 | 11.1 | 8.2 |
| 10,000 | 1.1 | 45.3 | 49.6 | 19.8 | 11.9 |
| 50,000 | 3.0 | 108.1 | 238.6 | 95.1 | 42.7 |
| \( T = 40 \) | | | |
| 1000 | 1.3 | 279.1 | 4.6 | 6.3 |
| 5000 | 2.6 | 698.9 | 12.9 | 15.7 |
| 10,000 | 4.3 | 1815.1 | 22.9 | 26.5 |
| 50,000 | 17.9 | 1472.8 | 31.7 | 115.5 |
it seems hard to justify the much higher error rates for the benefit of saving a few minutes of calculations.

The slowest method is the polynomial LDR, which for the high dimensional problem can take up to 25 min to find the solution and make the estimation of the out-of-sample distribution. This time is entirely dominated by the optimization—training—step, not the estimation—prediction—step. The polynomial LDR method runtime is extremely sensitive to the $p$ parameter. For example, for $T = 40$ and sample size 1000, it takes 33 s to solve with $p = 5$ and 279.1 s to solve with $p = 10$—the latter is the example shown in Table D.2.

The neural network model can be solved relatively fast, taking 2 min in the largest problem.

**APPENDIX E: SENSITIVITY ANALYSIS TO HYPER-PARAMETERS**

We discuss the sensitivity of our results with respect to the choice of hyper-parameters for the polynomial LDR and the neural network.

### E.1 Sensitivity of polynomial LDR

The polynomial LDR method in Section 4.4 has two hyper-parameters, the target dimensionality $p$ and the polynomial degree $\delta$. Additionally, the Riemannian BFGS algorithm used to solve the optimization problem adds several other parameters, the main one being the starting point for the parameter $A$, called here $A_0$.

The polynomial degree parameter is common to all polynomial approximations, and has the expected impact on the results. In this section, we focus on the parameter $p$, which is unique to the LDR and the parameter $A_0$, which in our empirical examples proved to have a large impact on results.

We show that the choice of $p$ is purely a trade-off between approximation error and number of samples required, and that the choice of starting point makes a very large difference in the final results. A random starting point performs relatively badly, compared to a starting point that takes into account the fact that in financial models, cash flows closer in time are usually more important than those farther in time.

**Starting point $A_0$**

The Riemannian BFGS algorithm used to solve the polynomial LDR optimization problem requires a starting point for $A$.

A first, simple way of generating a starting point—similar to what is done for the L-BFGS algorithm used to solve the neural network optimization problem—is to generate it randomly. To do this, we draw $dT \times p$ random samples from $N(0, 1)$ and arrange them into a $dT \times p$ matrix $B$. Then $A_0 = B(B^T B)^{-1/2}$ is a random matrix that follows the uniform distribution on the Stiefel manifold $V_p(\mathbb{R}^{dT})$.

A second way is to use a rectangular diagonal matrix and fill the last column to ensure that every one of the $dT$ input dimensions has a weight in at least one of the $p$ output dimension, that is $A_0 = B \mid B_{ij} = 1$ if $(i = j \land i \neq p) \land B_{i,p} = \frac{1}{\sqrt{dT-p+1}}$ if $i \geq p$. Conceptually, this starting point can be thought as a point where those input dimensions farthest in the future have been grouped into one output dimension. The following is an example for $dT = 4$ and $p = 3$:

$$
\begin{pmatrix}
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & \sqrt{0.5} \\
0 & 0 & \sqrt{0.5}
\end{pmatrix}.
$$

(E.1)
Table E.1 Insurance liability, comparison of expected shortfall MAPE (in percentage points) for different starting points in polynomial LDR basis

| Samples   | Folding | Diagonal | Random |
|-----------|---------|----------|--------|
| $T = 5$   |         |          |        |
| 1000      | 7.9     | 14.9     | 16.6   |
| 5000      | 5.3     | 17.5     | 31.5   |
| 10,000    | 5.1     | 23.6     | 31.8   |
| 50,000    | 3.7     | 31.0     | 32.4   |
| $T = 40$  |         |          |        |
| 1000      | 22.9    | 38.0     | 63.9   |
| 5000      | 5.7     | 54.7     | 69.4   |
| 10,000    | 5.9     | 53.0     | 68.2   |
| 50,000    | 7.2     | 49.2     | 67.0   |

A third way uses the same rationale of grouping input dimensions that are far in the future into one output dimension, but does so respecting the fact that $X \in \mathbb{R}^{T \times d}$ and only groups variables across time ($T$) but not across dimensions ($d$). The following is an example for $T = 5$, $d = 3$, and $p = 3$, which corresponds to what was used in the European call example in Section 5:

$$
\begin{pmatrix}
\sqrt{T} & 0 & 0 \\
0 & \sqrt{T} & 0 \\
0 & 0 & \sqrt{T} \\
\vdots & \vdots & \vdots \\
\sqrt{T} & 0 & 0 \\
0 & \sqrt{T} & 0 \\
0 & 0 & \sqrt{T}
\end{pmatrix}
$$

(E.2)

Since this method, which we call folding, provides the best results, we also use it in Sections 5 and 6. In the latter, we work with $T = 5$, $d = 5$, and $p = 10$, and leads to a starting point (in block notation):

$$
\begin{pmatrix}
\mathbf{1}_d & \mathbf{0}_{d \times (p-d)} \\
\mathbf{0}_{d \times d} & \sqrt{T - 1}\mathbf{1}_{p-d}
\end{pmatrix}^{T-1 \ times}.
$$

(E.3)

In Tables E.1 and E.2, which correspond to Tables 10 and 11, we can see the comparison across different starting points. The folding starting point performs best of all starting points. This is not surprising since it is expected that for this type of models—European call option and insurance liability—the combination of path dependency and discounting makes variables closer in time relatively more important than those farther in time. A disappointing characteristic revealed in the data is that when increasing the number of training samples, we do not always get a strictly decreasing error. In fact, the error seems to stabilize relatively early—around 5000 samples and then only be subject to small fluctuations. The comparison across starting points confirms that this lack of improvement is not due to a lack of a better solution, but rather most likely to the presence of local minima.
TABLE E.2 Insurance liability, comparison of relative mean $L_1$ error (in percentage points) for different starting points in polynomial LDR basis

| Samples | Folding | Diagonal | Random |
|---------|---------|----------|--------|
| T = 5   |         |          |        |
| 1000    | 0.2     | 1.0      | 1.2    |
| 5000    | 0.2     | 0.8      | 1.0    |
| 10,000  | 0.2     | 0.9      | 1.0    |
| 50,000  | 0.2     | 0.9      | 1.0    |
| T = 40  |         |          |        |
| 1000    | 0.8     | 1.7      | 1.7    |
| 5000    | 0.3     | 1.3      | 1.5    |
| 10,000  | 0.4     | 1.3      | 1.5    |
| 50,000  | 0.4     | 1.2      | 1.5    |

TABLE E.3 Insurance liability, comparison of expected shortfall MApE (in percentage points) for different values of the target dimensionality parameter in polynomial LDR basis

| Samples | Diagonal $p = 5$ | Diagonal $p = 10$ | Diagonal $p = 15$ | Diagonal $p = 20$ |
|---------|------------------|-------------------|-------------------|-------------------|
| T = 5   |                  |                   |                   |                   |
| 1000    | 22.3             | 14.9              | 228.0             | 767.9             |
| 5000    | 29.3             | 17.5              | 8.2               | 9.7               |
| 10,000  | 33.3             | 23.6              | 13.2              | 9.0               |
| 50,000  | 37.5             | 31.0              | 21.0              | 10.7              |

TABLE E.4 Insurance liability, comparison of relative mean $L_1$ error (in percentage points) for different values of the target dimensionality parameter in polynomial LDR basis

| Samples | Diagonal $p = 5$ | Diagonal $p = 10$ | Diagonal $p = 15$ | Diagonal $p = 20$ |
|---------|------------------|-------------------|-------------------|-------------------|
| T = 5   |                  |                   |                   |                   |
| 1000    | 1.0              | 1.0               | 3.1               | 37.3              |
| 5000    | 1.0              | 0.8               | 0.6               | 0.5               |
| 10,000  | 1.0              | 0.9               | 0.6               | 0.6               |
| 50,000  | 1.0              | 0.9               | 0.7               | 0.5               |

**Target dimensionality parameter $p$**

To show the effects of parameter $p$ on the insurance example, we choose one of the starting point methods (diagonal) and one maturity ($T = 5$). The results in Tables E.3 and E.4, which correspond to Tables 10 and 11, confirm the expected effect of changing this parameter: larger values of $p$ produce better results (since the feature map $\phi_\theta$ is a richer function) but also require more training samples to do so. We can see that when moving from $p = 10$—used in the main results for the insurance example—to $p = 15$ and therefore, from $m = 286$ to $m = 816$ the error for 1000 training samples increases by a factor of 10 in the expected shortfall and by a factor of 3 in the $L_1$ metric. In those cases with more training samples—5000 and above—the error goes down as expected.
TABLE E.5 Insurance liability, comparison of expected shortfall MApE (in percentage points) for different layer widths in neural network basis

| Samples | Fixed \( p = 100 \) | Minimum width | Equal param dims | Equal \( m \) |
|---------|-----------------|---------------|-----------------|--------------|
| \( T = 5 \) | | | | |
| 1000    | 2.9             | 2.2           | 3.1             | 3.9          |
| 5000    | 0.5             | 3.2           | 4.1             | 0.4          |
| 10,000  | 0.6             | 3.3           | 4.2             | 0.2          |
| 50,000  | 0.8             | 3.3           | 4.2             | 0.3          |
| \( T = 40 \) | | | | |
| 1000    | 14.4            | 14.5          | 17.5            | 14.1         |
| 5000    | 11.1            | 12.5          | 2.7             | 12.8         |
| 10,000  | 10.5            | 12.6          | 2.0             | 13.2         |
| 50,000  | 0.5             | 1.4           | 2.2             | 2.4          |

E.2 Sensitivity of neural network

The neural network method has one main hyper-parameter, the width of the hidden layer. Other typical neural network hyper-parameters as number of layers or activation function do not apply in this case, since the closed-form of the time-t expectation has been defined only for single-layer, ReLu networks. Unlike the polynomial LDR basis, we do not explore the impact of the starting point, since a random starting point already performs very well.

In the main results in Section 6, we use the same layer width in all cases, \( p = 100 \). This value is the results of a sensitivity test done for different values (10, 50, 100, 200) after which we chose the best results overall cases. This sensitivity test is similar to cross validation but is performed on entirely out-of-sample data, rather than partitioning the existing training data. This has the advantage of keeping the full size of the sample for each regression instead of having to reduce it to allow a percentage to be used as validation set. While cross validation is more frequently used when the total sample budget is fixed, sensitivity analysis is more adequate when one has the ability to generate as many out-of-sample sets as needed. In a different practical setting as the one in this paper, it might be more appropriate to use cross validation for the selection of hyper-parameters.

Using a single choice of layer width in all cases has the advantage of showing good overall results (for different maturities and training sample size) but the disadvantage of being neither optimized for each single case (meaning that the results could have been better when looking at each cell of the table) nor comparable to the polynomial method in terms of functional complexity, that is, the number of parameters that describe the feature map \( \phi_0 \).

The selection could have been done in different ways, and in this section, we show some alternatives and their effects on the results shown in Section 6. The results are summarized in Tables E.5 and E.6, which correspond to Tables 10 and 11. We show that some of the alternatives perform even better than our choice for the main results, implying the potential for improvement in the neural network basis, which is already the best performing basis in our comparisons.

The first alternative is to use the theoretical minimum width for the network, as described in Hanin and Sellke (2017). In our case, it means using \( p = 25 \) for \( T = 5 \) and \( p = 200 \) for \( T = 40 \). This method does not show a good performance. Interestingly, it performs worse even for \( T = 40 \) where \( p = 100 \) is below the minimum. Still, this is not a violation of the theoretical minimum since it
**TABLE E.6** Insurance liability, comparison of relative mean $L_1$ error (in percentage points) for different layer widths in neural network basis

| Samples  | Fixed $p = 100$ | Minimum width | Equal param dims | Equal $m$ |
|----------|-----------------|---------------|------------------|-----------|
| $T = 5$  |                 |               |                  |           |
| 1000     | 0.1             | 0.1           | 0.1              | 0.1       |
| 5000     | 0.1             | 0.1           | 0.1              | <0.1      |
| 10,000   | <0.1            | 0.1           | 0.1              | <0.1      |
| 50,000   | <0.1            | 0.1           | 0.1              | <0.1      |
| $T = 40$ |                 |               |                  |           |
| 1000     | 6.1             | 6.4           | 2.8              | 6.5       |
| 5000     | 0.6             | 0.8           | 0.5              | 0.8       |
| 10,000   | 0.5             | 0.5           | 0.2              | 0.5       |
| 50,000   | 0.1             | 0.1           | 0.1              | 0.1       |

assumes a neural network of arbitrary depth, so it is always possible that using more hidden layers would result in smaller errors than the fixed $p$ method.

The second alternative is to backsolve the width of the network that creates a parameter space of similar dimensionality as that of the polynomial LDR method. For $d = 5$, $T = 5$, and $p = 10$, the polynomial LDR has 481 parameters. For $T = 40$, it has 2231 parameters. This can be matched by using a neural network with $p = 18$ and $p = 11$ nodes, respectively. This alternative provides very good results for the high dimensionality case ($T = 40$) but not as good for the low dimensionality case ($T = 5$).

The third and final alternative is to use a neural network that matches the number of basis functions $m$. This means, for both $T = 5$ and $T = 40$, that $p = 286$. The results for this alternative are similar to the other alternatives.