A Machine Learning Approach to
Portfolio Pricing and Risk Management for
High-Dimensional Problems

Lucio Fernandez Arjona
University of Zurich

Damir Filipović
Ecole Polytechnique Fédérale de Lausanne and Swiss Finance Institute

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

Lucio Fernandez-Arjona[†]   Damir Filipović[‡]

11 March 2021

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.

keywords: Solvency capital; dimensionality reduction; neural networks; nested Monte Carlo; replicating portfolios.

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 the 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 behaviour 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 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 modelled 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. It is, however, global in the time di-

mension, unlike LSMC methods used for American option pricing, which are local in the time dimension. That is, the latter require repeated, recursive regressions. Within the LSMC family of methods, replicating martingales are a regress-later method ([GY02]) 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 financial derivative and an 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 and other LSMC methods, and we find our method offers superior results. Also in line with machine learning best practices, we publish the datasets for our examples, [FAF20]. We hope that these datasets can be used by others to allow more direct comparisons between methods in future research.

1.1 Related literature

An early example of static replication via basis functions can be found in [MM94], 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. [Car96] and [LS01] use a sequential approximation algorithm to calculate the conditional expectations required in the valuation of options with a 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 the method the name of LSMC. [AC03] apply this idea to valuation of life insurance policies.

The first distinction between regress-now and regress-later LSMC appears in [GY02]. 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 has interesting properties, among them less-dispersed estimates than regress-now LSMC.

While [GY02] described LSMC techniques in the context of the recursive regression approach used for the valuation of American options, our paper considers a broader definition—as does [PS16]—where regress-now methods are those that directly estimate the valuation function, and regress-later those that first estimate the terminal payoff of the contingent claim and based on that estimation they calculate the valuation function. 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 linear dimensionality reduction, 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, [BDM15] show that such regression-based methods can—asymptotically—improve the convergence rate of nested Monte Carlo methods. They provide quality comparisons against nested Monte Carlo, 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 [PS16]. 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 function. This function must be given to the method, and is based on expert judgement and knowledge of the problem domain. Moreover, the choice of this function implies a trade-off between complexity 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 [Duo19]. 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 [CFM+18]. 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 [HB20], 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 networks, for example [HJL17] and [RL18]. 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. We can see that there is a wider range of methods applied in a regress-now framework. Since regress-now methods estimate the conditional expectation function directly, it is possible to use almost any regression method in a regress-now framework. Implementing a regression method in a regress-later setting requires being able to calculate the conditional expectations separately, either in closed form or in another computationally efficient

manner. Additionally, regress-later methods have to deal with the time dimension of their inputs—which leads to larger-dimensional inputs—and must deal with path-dependency along that time dimension. In summary, they are more difficult to implement. In exchange, they are more accurate than their regress-now counterparts. It is, therefore, valuable to build regress-later methods with closed-form conditional expectations, which is what our paper contributes to the literature.

Table 1: *Comparison of regress-now and regress-later methods in the literature*

	Dimensionality reduction on inputs	Regression method
Regress-now		
[BDM15]	none	incomplete, manually-selected monomial basis
[Duo19]	none	polynomial spline space
[CFM+18]	implicit in neural network	neural network
[HB20]	none	left singular functions of the conditional expectation operator
[HJL17]	model decomposition	kernel smoothing
[RL18]	none	Gaussian process regression
Regress-later		
[PS16]	manually selected function	orthonormal piecewise linear basis and Hermite polynomials
this paper	linear dimensionality reduction	orthogonal polynomials and shallow neural networks

Another strand of related literature is in the field of uncertainty quantification, where polynomial surrogate functions have been used for a long time to reduce the runtime of complex models. Recently, [HC18] 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.

Nested Monte Carlo methods for portfolio management have been studied in [LG03] and [GJ10] among others. [GJ10] show several methods that can reduce the computational cost of nested Monte Carlo for a homogeneous portfolio of financial instruments with an additive structure. Since we aim to cover both financial derivatives and insurance portfolios—where it is common that portfolios do not exhibit such additive structure—we restrict our comparisons to standard nested Monte Carlo.

An unfortunate aspect of the literature is that there are no standard models on which to compare the quality of the surrogate models, as is the case in the field of machine learning with datasets like MNIST [LBBH98]. Such a dataset would allow more straightforward comparisons among the advanced methods mentioned above. In our paper we rely on a standard European call option example—which should be familiar to most readers—and an insurance model built for the purpose of this research. As in the large majority of the literature, we compare the replicating martingale approach to some well established methods—in our case nested Monte Carlo and regress-now LSMC—but not to any of the other advanced methods proposed in the literature. Our findings regarding the superior results of the replicating martingales have to be understood in the following

sense. These are empirical comparisons for a particular derivative and insurance example to two specific methods—nested Monte Carlo and regress-now LSMC. In this sense, we do not claim any general superiority of our approach.

The remainder of the paper is as follows. Section 2 formalizes the replicating martingale problem and recalls the standard nested Monte Carlo approach, which is then illustrated by means of a European call option example. Section 3 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 4 and 5 provide numerical case studies: the initial European call option example revisited in Section 4, and an insurance liability model in Section 5. Section 6 concludes. The appendix contains proofs and technical background material. Section A describes the quality metrics used to compare different methods. Section B contains the economic scenario generator underlying the numerical examples in the main text. Section C contains all proofs and some auxiliary results of independent interest. Sections D and E present 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. Randomness is generated by an $\mathbb{R}^d$-valued stochastic driver process $X = (X_1, \ldots, X_T)$ with mutually independent components X_t. 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, e.g., 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,

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

with $\mathcal{F}_t$-measurable time-t cash flows $\zeta_t = \zeta_t(X_1, \ldots, X_t)$. As is the case in practice, we assume that all ζ_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

$$V_t = \mathbb{E}^{\mathbb{Q}}_t[f(X)] = \underbrace{\sum_{s=1}^{t} \zeta_s}_{\text{accumulated cash flow at } t} + \underbrace{\mathbb{E}^{\mathbb{Q}}_t[\sum_{s=t+1}^{T} \zeta_s]}_{\text{time-}t \text{ spot value}},$$

where $\mathbb{E}^{\mathbb{Q}}_t[\cdot] = \mathbb{E}^{\mathbb{Q}}[\cdot \mid \mathcal{F}_t]$ denotes the $\mathcal{F}_t$-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, in-

surance 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*

$$\mathrm{VaR}_\alpha(Y) = \inf\left\{y : F_Y(y) \geq \alpha\right\} = F_Y^{-1}(\alpha)$$

and *expected shortfall* (also called conditional value at risk or average value at risk)

$$\mathrm{ES}_\alpha(Y) = \frac{1}{1-\alpha} \int_{1-\alpha}^1 \mathrm{VaR}_\gamma(Y) d\gamma$$

where Y is a random variable with $\mathbb{P}$-distribution function denoted by F_Y, and $\alpha \in (0,1)$ denotes the confidence level, see, e.g., [FS04, Section 4.4]. In insurance regulation, risk is usually measured for a one-year risk horizon and the economic capital is determined by $\rho[-\Delta V_1]$, where ρ is a placeholder for either VaR_α or ES_α. There are cases, however, where calculations require three-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 *nested Monte Carlo* 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 nested Monte Carlo 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.

2.1 Nested Monte Carlo for a European call option

By way of example, we calculate the present value (V_0) and expected shortfall of $-\Delta V_1$ 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 Section B. That generator maps X to a vector of economic factors. This vector contains several components, among them the equity index EQ_t
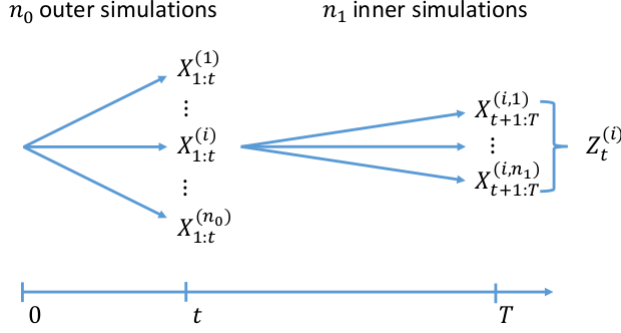
Figure 1: *Nested Monte Carlo structure. The outer simulations span the time period 0 to t, and each of those serves as the starting point of a group of inner simulations.*

and the cash account C_t. The European call payoff at time T is $\max(EQ_T - K, 0)$ where K is the strike of the option and T its maturity. The variables EQ_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

$$f(X) = -\max(EQ_T - K, 0)/C_T.$$

In this example we work with two maturities $T = 5$ and $T = 40$ and for both $K = 100$—that is, the option is at the money at time zero. The model has 3 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 nested Monte Carlo simulation, with 1,000,000 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 nested Monte Carlo 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 nested Monte Carlo estimators. The bias and variance of the nested Monte Carlo 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 Tables 2 and 3 for a fixed total budget of 50,000 simulations.[2] 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 Section 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 $\mathrm{ES}_{99\%}(-\Delta V_1)$.

---

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

[2] In 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.

8

Table 2: *Nested Monte Carlo (nMC), comparison of present value (50,000 total samples)*

Maturity	Benchmark	nMC	MApE
5	22.1691	22.1806	0.6%
40	67.5488	67.5567	1.0%

Table 3: *Nested Monte Carlo, comparison of expected shortfall for nested Monte Carlo (50,000 total samples)*

Maturity	Benchmark (value)	Nested Monte Carlo (MApE by inner simulations)							
		1	10	25	50	100	250	400	500
5	56.8588	233.4%	29.5%	11.8%	7.9%	9.3%	14.4%	19.0%	20.4%
40	62.6205	2027.0%	463.6%	232.1%	129.2%	66.6%	24.6%	21.4%	19.1%

As described in [BDM15], it is not possible, in general cases, to decide for a finite budget how to make this inner-outer split in an optimal way. In this paper we will err on the side of presenting optimistic risk figures for nested Monte Carlo estimations, by choosing an optimal split based on our knowledge of the benchmark value. This bias towards more accurate nested Monte Carlo 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 nested Monte Carlo, which is already better than what one would obtain in practice. Since the optimal number of simulations for risk calculations is different than for the pricing calculations, it is not possible to be optimal for both at the same time. We have chosen the optimal set for risk calculations. This problem does not exist for regression-based methods, since it is not necessary to split the training budget.

The effect on the present value calculations of using the split which is for optimal for risk calculations can be seen in Table 4. There, for example, the error on the 5 year call is 1.7% while in Table 2 we can see that the error would be 0.6% when using the split optimal for present value calculations—which corresponds to flat, not nested, Monte Carlo.

Table 4: *Nested Monte Carlo, comparison of present value and expected shortfall MApE (in percentage points) for optimal inner–outer split*

Samples	Present Value		Expected Shortfall	
	Maturity: 5	Maturity: 40	Maturity: 5	Maturity: 40
1,000	6.5	7.5	26.7	403.6
5,000	4.0	3.9	15.9	110.9
10,000	3.8	3.1	14.5	56.6
50,000	1.7	2.4	7.9	19.1

In order to select the optimal outer-inner combination, each combination is run 100,000 times to be able to estimate its MApE ES. The combination with the lowest MApE ES is chosen as the optimal for that (total) sample size. Table 4 shows the relationship between the MApE ES of the optimal combination and the total sample size. We can see that the estimation of $\mathrm{ES}_{99\%}[\Delta V_1]$,

performed via nested Monte Carlo, is much more affected by the increased dimensionality of the problem than the estimation of V_0.

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 nested Monte Carlo 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.

3 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 $L_{\mathbb{Q}}^2$ 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.

3.1 Finite-dimensional approximation

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

$$\min_{\beta \in \mathbb{R}^m} \|f - \phi_\theta^\top \beta\|_{L_{\mathbb{Q}}^2}. \tag{1}$$

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

$$G_{\theta,t}(x_1, \ldots, x_t) = \mathbb{E}^{\mathbb{Q}}[\phi_\theta(x_1, \ldots, x_t, X_{t+1}, \ldots, X_T)], \tag{2}$$

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

$$V_{\theta,t} = G_{\theta,t}(X_1, \ldots, X_t)^\top \beta_\theta, \tag{3}$$

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 [GY02]. As we will see in the numerical examples below, this approach performs much better than the alternative

regress now.

So, how good is our approximation (3)? By Doob's inequality, making use of the martingale property of the value process, we obtain an upper bound on the pathwise maximal $L_{\mathbb{P}}^1$-error,

$$\|\max_{t \leq T} |V_t - V_{\theta,t}|\|_{L_{\mathbb{P}}^1} \leq \|\tfrac{d\mathbb{P}}{d\mathbb{Q}}\|_{L_{\mathbb{Q}}^2} \|\max_{t \leq T} |V_t - V_{\theta,t}|\|_{L_{\mathbb{Q}}^2} \leq 2\|\tfrac{d\mathbb{P}}{d\mathbb{Q}}\|_{L_{\mathbb{Q}}^2}\|f - f_\theta\|_{L_{\mathbb{Q}}^2}. \tag{4}$$

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

3.2 Feature learning

We now learn the parameter θ 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 θ, which leads to the non-convex optimization problem

$$\min_{(\theta,\beta) \in \Theta \times \mathbb{R}^m} \|f - \phi_\theta^\top \beta\|_{L_{\mathbb{Q}}^2}. \tag{5}$$

Here is an elementary existence result.

Lemma 1. *Assume*

(i) Θ *is compact,*

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

(iii) $\{\phi_{\theta,1}, \dots, \phi_{\theta,m}\}$ *is linearly independent in* $L_{\mathbb{Q}}^2$ *for all* $\theta \in \Theta$.

Then there exists a solution to (5).

The assumptions in Lemma 1 cannot be relaxed in general. This is shown by the following example.

Example 1. *Let* $\mathbb{Q} = \frac{1}{2}\sum_{i=1}^2 \delta_{x^{(i)}}$ *be a discrete measure supported on two points* $x^{(1)} \neq x^{(2)}$, *so that every function* $f \in L_{\mathbb{Q}}^2$ *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* $\phi_\theta = \frac{(1_{(0,1]}(\theta),\theta+1_{\{0\}}(\theta))}{\sqrt{1+\theta^2}}$, *so that* $\phi_0 = (0,1)$ *(not continuous), or*

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

For either case, we have $\inf_{(\theta,\beta)\in\Theta\times\mathbb{R}} \|f - \phi_\theta\beta\|_{\mathbb{R}^2} = \lim_{\theta\to 0} \|f - \phi_\theta\beta_\theta\|_{\mathbb{R}^2} = 0$, *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 (5) 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 linear dimensionality reduction, as follows. We henceforth assume that the feature map is parametrized in the form

$$\phi_{\theta,i}(x) = g_i(A^\top x + b) \tag{6}$$

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 ϕ_θ at the $\mathbb{R}^{d\times T}$-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 $A^\top = (A_1^\top, \ldots, A_T^\top)$. Then we have $A^\top \mathrm{vec}(X) = \sum_{t=1}^T A_t^\top X_t$ and

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

The conditional expectation functions in (2) read component-wise for $i = 1, \ldots, m$ as

$$G_{\theta,t,i}(x_1, \ldots, x_t) = \mathbb{E}^{\mathbb{Q}}[g_i(b + \textstyle\sum_{s=1}^t A_s^\top x_s + \sum_{s=t+1}^T A_s^\top X_s)]. \tag{7}$$

We remark that our approach does not replace the original stochastic driver X by $A^\top X + b$ in general, only for the specific portfolio f to which it is calibrated through (5). In the sequel, we study three different specifications of the type (6) in more detail: a full polynomial basis, a polynomial feature map with linear dimensionality reduction, and a shallow neural network.

3.3 Full polynomial basis

We start with the non-weighted and non-biased 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 (5) boils down to the projection (1). We let the feature map ϕ be composed of a basis of the space of all polynomials of degree δ or less,

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

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

$$\mathbb{E}_{\mathbb{Q}}[\|X\|^{2\delta}] < \infty. \tag{8}$$

No attempt is made at selecting individual basis elements from within $\mathrm{Pol}_\delta(\mathbb{R}^{dT})$ at this stage,

every polynomial is used in the projection. As a consequence—albeit tractable, as we shall see 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,

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

Table 5 shows that the dimension m quickly becomes larger than the training sample size in practice.[3]

T	$d = 3$	$d = 5$
5	816	3,276
40	302,621	1,373,701

Table 5: *Dimension $m = \dim \text{Pol}_\delta(\mathbb{R}^{dT})$ for $\delta = 3$.*

The exact form of the conditional expectation (2) depends on the choice of ϕ and the distribution $\mathbb{Q} = \otimes_{t=1}^T \mathbb{Q}_t$ of X. Choosing an orthogonal basis of polynomials in $L_{\mathbb{Q}}^2 = \otimes_{t=1}^T L_{\mathbb{Q}_t}^2$ can greatly simplify the calculations. In view of [Sul15, Theorem 8.25], there is a system of orthogonal polynomials $\{g_{\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{N}_0^{dT}\}$ on $\mathbb{R}^{dT}$ for $\mathbb{Q}$ that can be expressed as

$$g_{\boldsymbol{\alpha}}(x) = \prod_{t=1}^T h_{t,\boldsymbol{\alpha}_t}(x_t), \quad x = \text{vec}(x_1,\ldots,x_t), \quad \boldsymbol{\alpha} = (\boldsymbol{\alpha}_1,\ldots,\boldsymbol{\alpha}_T),$$

where $\{h_{t,\boldsymbol{\alpha}} \mid \boldsymbol{\alpha} \in \mathbb{N}_0^d\}$ is a system of orthogonal polynomials on $\mathbb{R}^d$ for $\mathbb{Q}_t$, for every $t = 1,\ldots,T$. Moreover, $\deg g_{\boldsymbol{\alpha}} = |\boldsymbol{\alpha}|$ and $\deg h_{t,\boldsymbol{\alpha}} = |\boldsymbol{\alpha}|$. Now choose an index mapping [4] $\{1,\ldots,m\} \ni i \mapsto \boldsymbol{\alpha}^i \in \{\boldsymbol{\alpha} \in \mathbb{N}_0^{dT} \mid |\boldsymbol{\alpha}| \leq \delta\}$, then we obtain an orthogonal basis of $\text{Pol}_\delta(\mathbb{R}^{dT})$ for $\mathbb{Q}$ by setting $\phi_i = g_{\boldsymbol{\alpha}^i}$. The conditional expectation functions (7) are then given in closed form as

$$G_{t,i}(x_1,\ldots,x_t) = \prod_{s=1}^t h_{s,\boldsymbol{\alpha}_s^i}(x_s) \prod_{s=t+1}^T 1_{\boldsymbol{\alpha}_s^i = \mathbf{0}}. \tag{9}$$

This extends to the unconditional expectations, $\mathbb{E}^{\mathbb{Q}}[\phi_i(X)] = \phi_i(0)1_{\boldsymbol{\alpha}^i=\mathbf{0}}$.

Example 2. *Consider the multinormal case $\text{vec}(X) \sim N(0, I_{dT})$. Here we can choose $g_{\boldsymbol{\alpha}}$, and $h_{t,\boldsymbol{\alpha}} \equiv h_{\boldsymbol{\alpha}}$, as the multivariate (probabilists') Hermite polynomials of order $\boldsymbol{\alpha} \in \mathbb{N}_0^{dT}$ on $\mathbb{R}^{dT}$, and order $\boldsymbol{\alpha} \in \mathbb{N}_0^d$ on $\mathbb{R}^d$, respectively.*

---

[3]Strictly speaking, the dimension of the linear span of the functions ϕ_i in $L_{\mathbb{Q}}^2$ could be less than m, because they may be linearly dependent as elements in $L_{\mathbb{Q}}^2$. This is in particular the case when $\mathbb{Q}$ is an empirical measure from a sample of size n, as described in Subsection 3.6. In this case, $m > n$ would simply lead to an exact interpolation of f, which likely will result in overfitting.

[4]This 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.

3.4 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 $\mathrm{Pol}_\delta(\mathbb{R}^p)$, and we consider all feature maps (6) for weight matrices A with full rank and bias vectors b. As above, we assume that (8) holds, so that $\phi_{(A,b),i} \in L_\mathbb{Q}^2$. 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 1. *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 $\mathrm{span}\{\phi_{(A,b),1}, \ldots, \phi_{(A,b),m}\} = \mathrm{span}\{\phi_{(\tilde{A},0),1}, \ldots, \phi_{(\tilde{A},0),m}\}$,*

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

In view of Theorem 1 the parameter set can be chosen to be $\Theta = V_p(\mathbb{R}^{dT})$ yielding feature maps of the form

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

without loss of generality. We arrive at the following existence and non-uniqueness result.

Theorem 2. *For the polynomial feature map, there exists a minimizer in $\Theta = V_p(\mathbb{R}^{dT})$ of (5). However, uniqueness does not hold, in the sense that the optimal subspace $\mathrm{span}\{\phi_{A,1}, \ldots, \phi_{A,m}\}$ is not unique, in general.*

Problem (5) corresponds to a *linear dimensionality reduction* with matrix manifold $V_p(\mathbb{R}^{dT}) \times \mathbb{R}^m$ in the spirit of [CG15]. The dimensionality reduction is produced exclusively by the linear mapping A of $\mathbb{R}^{dT}$ onto $\mathbb{R}^p$. There is no other restriction imposed on $\mathrm{Pol}_\delta(\mathbb{R}^p)$, every polynomial basis function g_i is used. And yet, the dimensionality reduction compared to the full polynomial basis of $\mathrm{Pol}_\delta(\mathbb{R}^{dT})$ is significant. Indeed, the total dimension of the optimization problem (5) is given by the sum of $\dim V_p(\mathbb{R}^{dT}) = dTp - \frac{1}{2}p(p+1)$ plus $m = \dim \mathrm{Pol}_\delta(\mathbb{R}^p)$. This sum can be kept low by choosing p small enough. Table 6 shows that the total dimension of (5) remains moderate compared to the corresponding figures of the full polynomial basis from Table 5.

T	$d = 3,\ p = 3$	$d = 5,\ p = 10$
5	$9 + 20 = 29$	$195 + 286 = 481$
40	$114 + 20 = 134$	$1{,}945 + 286 = 2{,}231$

Table 6: *Total dimension $\dim V_p(\mathbb{R}^{dT}) + \dim \mathrm{Pol}_\delta(\mathbb{R}^p)$ for $\delta = 3$.*

The calculation of the conditional expectation (2) is not as simple as for the full polynomial basis in (9). Instead we need to compute the unconditional moments (7), which here reduce to

$$G_{A,t,i}(x_1, \ldots, x_t) = \mathbb{E}^\mathbb{Q}[g_i(\textstyle\sum_{s=1}^t A_s^\top x_s + \sum_{s=t+1}^T A_s^\top X_s)], \tag{10}$$

for $A \in V_p(\mathbb{R}^{dT})$. Evaluation of (10) 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 [Kan08, Lemma 1], which

14

relates multivariate moments to univariate moments, generalizing $4y_1y_2 = (y_1 + y_2)^2 - (y_1 - y_2)^2$, by

$$y^{\boldsymbol{\alpha}} \equiv y_1^{\alpha_1} \cdots y_p^{\alpha_p} = \frac{1}{|\boldsymbol{\alpha}|!} \sum_{\boldsymbol{\nu}=\mathbf{0}}^{\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\nu}|} \binom{\alpha_1}{\nu_1} \cdots \binom{\alpha_p}{\nu_p} (h_{\boldsymbol{\alpha},\boldsymbol{\nu}}^\top y)^{|\boldsymbol{\alpha}|}, \tag{11}$$

for vectors $h_{\boldsymbol{\alpha},\boldsymbol{\nu}} = (\alpha_1/2 - \nu_1, \ldots, \alpha_p/2 - \nu_p)^\top$. The sum in (11) 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 (10) reduces to the calculation of the $|\boldsymbol{\alpha}|$th moments of the scalar random variables $h_{\boldsymbol{\alpha},\boldsymbol{\nu}}^\top Y$, which are given in closed form for various distributions of X.

Example 3. *For the multinormal case* $\text{vec}(X) \sim N(0, I_{dT})$ *we have*

$$h_{\boldsymbol{\alpha},\boldsymbol{\nu}}^\top Y \sim N(\textstyle\sum_{s=1}^t h_{\boldsymbol{\alpha},\boldsymbol{\nu}}^\top A_s^\top x_s, \sum_{s=t+1}^T \|A_s h_{\boldsymbol{\alpha},\boldsymbol{\nu}}\|^2).$$

These univariate moments are given in closed form, as explicitly stated in [Kan08, Proposition 2].

3.5 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 (6) of the form

$$\phi_{(A,b),i}(x) \equiv \phi_{(a_i,b_i)}(x) = (a_i^\top x + b_i)^+$$

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 (8) holds for $\delta = 1$, so that $\phi_{(a_i,b_i)} \in L_{\mathbb{Q}}^2$. By the positive homogeneity of the components of the feature map in the parameter, $\phi_{(\lambda a_i, \lambda b_i)} = \lambda \phi_{(a_i,b_i)}$ 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 $\phi_{(a_i,b_i)}$? Here is a fundamental result, which seems to be little known in the literature.

Theorem 3. *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*

$$\text{span}\{\phi_{(a_1,b_1)}, \ldots, \phi_{(a_m,b_m)}\} = \text{span}\{\phi_{(\tilde{a}_1,\tilde{b}_1)}, \ldots, \phi_{(\tilde{a}_m,\tilde{b}_m)}\} \tag{12}$$

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*

$$\{\phi_{(a_1,b_1)}, \ldots, \phi_{(a_m,b_m)}\} \text{ *is linearly independent.*} \tag{13}$$

Note that the converse implication in Theorem 3(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

15

example shows that the assumptions in Theorem 3(ii) cannot be relaxed to pairwise inequality, $(a_i, b_i) \neq (a_j, b_j)$ for all $i \neq j$.

Example 4. *Let $(a_1, b_1), \ldots, (a_3, b_3) \in S_{dT}$ be linearly dependent vectors such that $\sum_{i=1}^{3} c_i(a_i, b_i) = 0$ 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 $\{\phi_{(a_i, b_i)} \mid i = 1, \ldots, 6\}$ is linearly dependent,*

$$\textstyle\sum_{i=1}^{3} c_i \phi_{(a_i, b_i)} + \sum_{i=1}^{3} (-c_i) \phi_{(a_{3+i}, b_{3+i})} = \sum_{i=1}^{3} c_i (a_i^\top x + b_i) = 0,$$

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

We conclude that the assumptions of the existence Lemma 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. *For the shallow ReLU neural network, there exists no minimizer of (5) in general. Moreover, uniqueness does not hold, in the sense that the optimal subspace $\mathrm{span}\{\phi_{(a_1, b_1)}, \ldots, \phi_{(a_m, b_m)}\}$ is not unique, in general.*

The conditional expectation functions (7) read here as

$$G_{(a_i, b_i), t}(x_1, \ldots, x_t) = \mathbb{E}^{\mathbb{Q}}[(b_i + \textstyle\sum_{s=1}^{t} a_{i,s}^\top x_s + \sum_{s=t+1}^{T} a_{i,s}^\top X_s)^+], \tag{14}$$

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 (14) boils down to compute $\mathbb{E}^{\mathbb{Q}}[Y^+]$ 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 5. *For the multinormal case $\mathrm{vec}(X) \sim N(0, I_{dT})$, we have*

$$Y \sim N(b_i + \textstyle\sum_{s=1}^{t} a_{i,s}^\top x_s, \sum_{s=t+1}^{T} \|a_{i,s}\|^2). \tag{15}$$

We then obtain a closed form expression for (14) by combining (15) with the well known Bachelier's call option price formula $\mathbb{E}^{\mathbb{Q}}[Z^+] = \mu \Phi(\mu/\sigma) + \sigma \Phi'(\mu/\sigma)$, for a normal distributed random variable $Z \sim N(\mu, \sigma^2)$, where Φ denotes the standard normal distribution function and Φ' its density function, see, e.g., [DS06, Section 4.3] or [FA21].

In other cases, where the extended Fourier transforms $\widehat{\mathbb{Q}}_t(u) = \mathbb{E}_{\mathbb{Q}}[e^{u^\top X_t}]$ of the marginal distributions $\mathbb{Q}_t$ 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

$$y^+ = \tfrac{1}{2\pi} \int_{\mathbb{R}} e^{(w+i\lambda)y} \tfrac{1}{(w+i\lambda)^2} \, d\lambda.$$

Hence the evaluation of (14) reduces to the computation of the line integral

$$G_{(a_i, b_i), t}(x_1, \ldots, x_t) = \tfrac{1}{2\pi} \int_{\mathbb{R}} \widehat{F}_Y(w + i\lambda) \tfrac{1}{(w+i\lambda)^2} \, d\lambda, \tag{16}$$

where $\widehat{F}_Y(w + \mathrm{i}\lambda) = \mathbb{E}_\mathbb{Q}[\mathrm{e}^{(w+\mathrm{i}\lambda)Y}] = \mathrm{e}^{(w+\mathrm{i}\lambda)(b_i + \sum_{s=1}^{t} a_{i,s}^\top x_s)} \prod_{s=t+1}^{T} \widehat{\mathbb{Q}}_s((w+\mathrm{i}\lambda)a_{i,s})$ is in closed form. Note that Fourier type integrals like the one in (16) are routinely computed in finance applications, e.g, in Lévy type or affine models, [DFS03]. So one can draw on existing libraries of computer code.

3.6 Finite-sample estimation

While surprising and remarkable, the non-uniqueness and non-existence results in Theorems 2 and 4 for polynomial feature maps with dimensionality reduction and shallow ReLU neural networks, respectively, are mainly of theoretical interest, arguably. In practice, we solve (5) numerically using some quasi-Newton algorithm, which finds local minima that serve as approximate solutions. Thereto, we replace the model population measure $\mathbb{Q}$ by the empirical measure $\widehat{\mathbb{Q}} = \frac{1}{n} \sum_{i=1}^{n} \delta_{x^{(i)}}$ based on a training sample $x^{(1)}, \ldots, x^{(n)}$ drawn from $\mathbb{Q}$, along with the corresponding function values $y_i = f(x^{(i)})$.

For the full polynomial basis, problem (5) boils down to the projection (1), and we obtain the optimal

$$\widehat{\beta} = (\tfrac{1}{n}\Phi^\top\Phi)^{-1}(\tfrac{1}{n}\Phi^\top y) \tag{17}$$

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 $\widehat{\beta}$ converges in probability to the optimal β in (1) for the model population measure $\mathbb{Q}$, as the sample size $n \to \infty$. Moreover, the central limit theorem holds and theoretical guarantees for the sample error can be established, see, e.g., [BF19].

For the polynomial feature map with dimensionality reduction, we use the Riemannian BFGS algorithm [HGA15] to find a local minimizer of (5) 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 [PVG$^+$11] for the Python programming language to find a local minimizer of (5) 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.

4 European call option example revisited

Having presented the theoretical background, we now turn back to our initial illustrating example that we introduced in Section 2.1. We apply the functional bases described in Sections 3.3, 3.4, and 3.5, following the steps outlined in Section 3.6. For each functional basis we show the same quality metrics as in Section 2.1 in order to compare to the results from nested Monte Carlo 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 ΔV_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 Section A.

4.1 Results

The first comparison uses the full polynomial basis described in Section 3.3. In Tables 7 and 8 we present the MApE comparison among nested Monte Carlo (nMC), regress-now polynomial basis and the replicating martingale full polynomial basis. Table 7 shows results for the present value and Table 8 for the expected shortfall estimators. 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 9. We can see that Table 8 confirms the conclusions from Table 9, 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.

Table 7: *European call, comparison of present value MApE (in percentage points)*

| Samples | nMC | Full Polynomial basis | | LDR | Neural Network | |
		Regress-now	Regress-later	Regress-later	Regress-now	Regress-later
T=5						
1,000	6.5	4.0	1.3	0.5	4.2	0.2
5,000	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						
1,000	7.5	7.2		3.8	7.9	5.5
5,000	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 8: *European call, comparison of expected shortfall MApE (in percentage points)*

| Samples | nMC | Full Polynomial basis | | LDR | Neural Network | |
		Regress-now	Regress-later	Regress-later	Regress-now	Regress-later
T=5						
1,000	26.7	21.5	4.8	2.0	47.5	0.9
5,000	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						
1,000	403.6	141.0		10.0	459.0	16.0
5,000	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

Tables 7 and 8 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

Table 9: *European call, comparison of relative mean L_1 error (in percentage points)*

	Full Polynomial basis		LDR	Neural Network	
Samples	Regress-now	Regress-later	Regress-later	Regress-now	Regress-later
T=5					
1,000	15.5	4.6	1.1	36.6	0.5
5,000	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					
1,000	26.5		4.6	68.8	7.4
5,000	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

functions as the dimensionality of the problem grows. As shown in Table 5, 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 linear dimensionality reduction (LDR), as described in Section 3.4. Table 6 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 linear dimensionality reduction for a small δ and p depends on the nature of the function. Asymptotically, any function in $L_{\mathbb{Q}}^2$ can be approximated with arbitrary precision.

The optimization problem in (5) 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 1. This reduces the effective dimensionality of the problem and simplifies the calculation of the conditional expectation of $\phi(X)$ in Equation (3). For these examples we have used the Riemannian BFGS algorithm from [HGA15] using the C++ library published by the authors. Additionally, we tested another two algorithms, Riemannian Trust Regions ([ABG07]) as implemented by the Python library *pymanopt* ([TKW16]), and Grassmann Gauss-Newton ([HC18]) as implemented by the authors in the publicly available Python library. In all cases, Riemannian BFGS achieved better results.

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 Section D, and a description of the methodology for the sensitivity analysis is found in E. In summary, p is an important hyperparameter but it is not special in any way, which means that any robust method for selecting hyperparameters—cross-validation, pilot simulation, sensitivity analysis—can be used.

In Section D, 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 7 and 8 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 nested Monte Carlo 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 3.5. 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 $1{,}600 + 101 = 1{,}701$ 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 ([PVG$^+$11]). The results in Tables 7 and 8 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 (1,000).

In the comparison between neural network approaches, we can see that the replicating martingale—a regress-later method—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 nested Monte Carlo 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 non-linear 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 (6), both methods use a linear map to reduce the dimensionality of the input before applying a non-linear 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.

4.2   Comparison to lasso

The lasso ([Tib96]) is a popular model building technique for generating sparse models. It is used very successfully as a method for feature selection, especially in high dimensional problems. For that reason it is natural to compare the lasso regressor with the replicating martingale approach
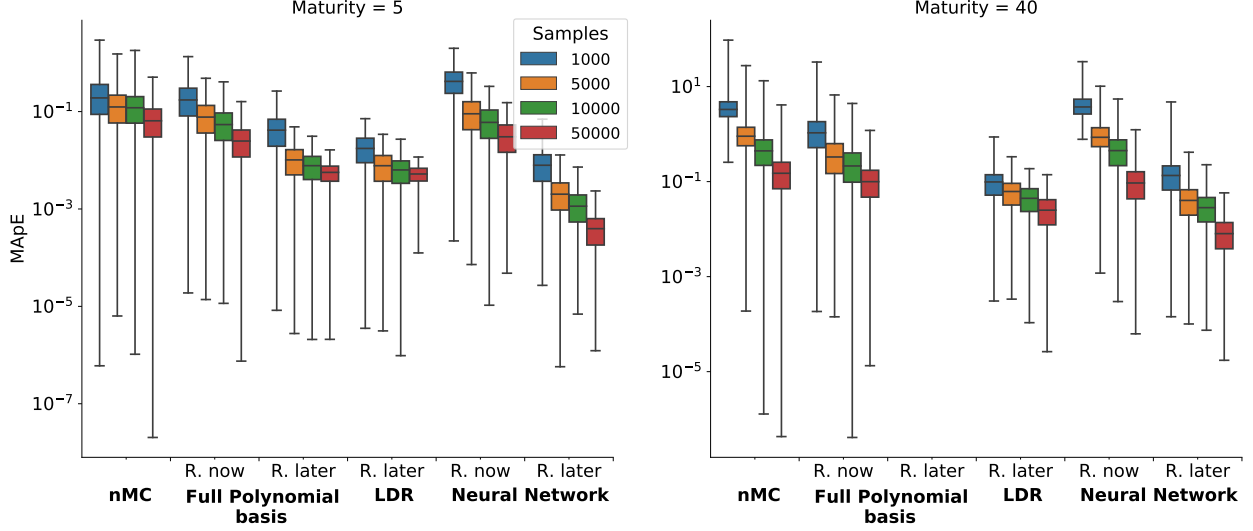
Figure 2: *Distribution of expected shortfall estimates per method for the European call example. Boxes show the upper and lower quartiles of the empirical distributions, while whiskers show their maxima and minima. Due to the logarithmic scale, a visual artifact is introduced by which the inter-quartile 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.*

we present in this paper. At the same time, the comparison reveals the interesting differences in the way that replicating martingales and lasso perform the dimensionality reduction.

The lasso estimates of a linear model on a full polynomial basis, as in Section 3.3 are obtained by

$$\widehat{\beta} = \arg\min_{\beta} \|y - \Phi^{\top}\beta\|^2_{\mathbb{R}^n} + \lambda \sum_{j=1}^{m} |\beta_j|$$

where Φ is defined as in (17).

One important difference is that lasso does not change the size of the regression problem, only the sparsity of the solution. Therefore the regression problem does not become smaller, unlike the case of polynomials with linear dimensionality reduction where once the input variables are projected, the remaining OLS problem is much smaller than the original.

A second important difference is that lasso selects among preexisting features, rather than creating new ones. The methods we described above create new features adaptively, under the assumption that the true function can be expressed in a low dimensional space. This lower dimension number p must be specified and becomes fixed, which is a disadvantage in respect with lasso which does not require a predetermined number of features. The ability to *create* new features gives the replicating martingale an advantage over lasso, which is limited to the input feature map. However, doing so while being faster in high-dimensional cases is part of what makes the replicating martingales an interesting method. The next section contains a comparison of runtime across different methods.

For this numerical example, the lasso estimator is built from the full polynomial basis—816

elements for $T = 5$ and $302,621$ for $T = 40$. The feature map used for lasso is consequently the upper bound of the polynomial LDR feature map, that is, the one corresponding to $p = 15$ and $p = 120$ for $T = 5$ and $T = 40$ respectively. In the example we use $p = 3$, and therefore the success of the polynomial LDR estimator will depend on whether a good solution can be expressed in the lower dimensional space of the resulting feature map.

To automate the selection of the penalty factor, we use the Akaike information criterion, using the algorithm provided by scikit-learn in its LassoLarsIC module, which is partly based on [ZHT$^+$07].

Table 10 shows the comparison between lasso and two replicating martingale methods—LDR and neural network—for both MApE and L_1 error. The quality of lasso is consistently worse than that of the other estimators, both in the tail and across the body of the distribution.

Table 10: *European call, comparison of MApE ES and relative mean L_1 error (both in percentage points)*

Samples	MApE ES			Rel. mean L_1 error		
	Lasso	LDR	N. Net.	Lasso	LDR	N. Net.
T=5						
1,000	24.1	2.0	0.9	8.4	1.1	0.5
5,000	8.8	0.9	0.2	2.6	0.6	0.4
10,000	7.3	0.7	0.1	2.2	0.6	0.4
50,000	1.7	0.5	0.0	0.9	0.5	0.4
T=40						
1,000	100.0	10.0	16.0	64.2	4.6	7.4
5,000	83.9	6.5	4.9	24.7	2.6	2.2
10,000	70.8	4.9	3.3	17.9	2.0	1.4
50,000		2.9	1.0		1.3	0.8

4.3 Comparison of runtime

The last comparison that we present on this European call example shows how long it takes to run the training and prediction phases on each model. This involves: running the regression on the number of samples indicated on the first column and calculating $\widehat{V}_1(x_i)$ for 1,000,000 validation samples.

The results are presented in Table 11. They clearly show the effect of the dimensionality reduction in the computational cost of the replicating martingale method. In high-dimensional problems, where the lasso regression must work with all features in the high-dimensional space, the LDR and Neural Network models are two orders of magnitude faster since the dimensionality reduction takes place prior to the creation of the features, thus creating a smaller regression problem.

Table 11: *European call, comparison of runtime (in seconds), single core AMD Opteron 6380*

Samples	Lasso	Full Polynomial basis		LDR	Neural Network	
		Regress-now	Regress-later	Regress-later	Regress-now	Regress-later
T=5						
1,000	1.7	0.9	1.5	1.4	9.5	18.4
5,000	3.4	0.9	2.3	1.6	16.1	21.1
10,000	6.1	0.9	3.0	2.1	24.0	24.4
50,000	24.9	1.0	13.4	7.1	92.6	49.4
T=40						
1,000	153.4	0.9		7.2	10.0	18.5
5,000	897.2	0.9		22.3	15.5	24.4
10,000	2,199.0	1.0		51.7	24.0	30.0
50,000		1.4		271.7	19.1	76.1

5 Insurance liability model example

Having shown the effectiveness of learning the replicating martingale in the case of a European call, we present now a much 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.

The example is constructed as follows. There are five stochastic drivers, $d = 5$: 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 are those described in Section B and used in previous examples. The real estate model is the same as the equity model from Section B, but uses an independent stochastic driver and a lower volatility than the equity model. The stochastic mortality follows a Lee–Carter model ([LC92]) to provide a trend and random fluctuations over time.

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 fix asset allocation, 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 ten-year zero coupon bond, a twenty-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

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

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

$$
A_t = \sum_{i=1}^{4} U_t^i V_t^i = \sum_{i=1}^{4} U_{t-1}^i V_t^{i-} + P_t,
$$

where

$$
V_t^i = \begin{cases} B(t, t+10), & i = 1, \\ B(t, t+20), & i = 2, \\ EQ_t, & i = 3, \\ RE_t, & i = 4, \end{cases} \quad \text{and} \quad V_t^{i-} = \begin{cases} B(t, t+9), & i = 1, \\ B(t, t+19), & i = 2, \\ V_t^i, & i = 3, 4, \end{cases}
$$

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

- $U_t^i = (U_{t-1}^i V_t^{i-} + M_i P_t)/V_t^i$: number of units of asset i held in period $(t, t+1]$ (per policy), where $U_{-1}^i := 0$

- $B(t, S)$: value at time t of a bond maturing at time S

- EQ_t: value of equity index at time t

- RE_t: value of real estate at time t

- M_i: asset allocation mix, henceforth fixed to $M = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{5}, \frac{2}{15} \right)$

- P_t: premium paid at t for period $(t, t+1]$ (per policy)

24

Bond prices (B), equity and real estate indices (EQ and RE) are simulated according to the model in Section B.

The policy variables are given by

$$P_t = 100, \qquad \text{and} \qquad G_t = \begin{cases} 0, & t = 0 \\ G_{t-1} + P_{t-1}, & t \geq 1 \end{cases}$$

As for the demographic variables, the total dead in period $(t-1, t]$ and total alive at t are given by

$$D_t = \sum_x D_t^x, \qquad \text{and} \qquad L_t = \sum_x L_t^x,$$

where

- $D_t^x = L_{t-1}^x q_x(t)$: total dead of age x at $t-1$ in period $(t-1, t]$

- $L_t^x = L_{t-1}^x - D_t^x$: 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 $(t-1, t]$.

The risk factor underlying the demographic variables is the death rate, which we model with the Lee–Carter mortality model

$$q_x(t) = 1 - e^{-m_x(t)} \qquad\qquad m_x(t) = e^{a_x + b_x k(t)}$$
$$k(t) = k(t-1) - 0.365 + \epsilon_t \qquad\qquad \epsilon_t = 0.621 X_t^{(lc)}$$
$$k(0) = -11.41$$

where

- $m_x(t)$: force of mortality at time t for age x

- $X_t^{(lc)}$: 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)

Before we present our numerical results, as an interlude, we now discuss in the high dimensionality of this example.

5.1 High dimensionality in insurance models

In a world without limitations to sample size, the existence of high-dimensional models is not a concern. Since this paper attempts to present a practical methodology, it is important to discuss what *high-dimensional* means in this context.

Solvency models have two sources of dimensionality: the outer simulations—the real-world distribution—and the inner simulations—the risk-neutral distribution. The real-world distribution

usually has hundreds or thousands of variables in models currently used in the insurance industry. The risk-neutral distribution—which is used to price the liabilities—has far fewer dimensions. In [ADGA08] the authors describe the scenario generator developed jointly by the Casualty Actuarial Society and the Society of Actuaries. The model contains five stochastic drivers. This means that no matter how many stochastic drivers the real-world distribution has, the calculation of the time-t conditional expectations of the asset–liability portfolio will only consider five of those stochastic drivers. Is this a problem? In our professional practice, we have observed that while the full economic model might have dozens of equity risk drivers—for example, for different countries—any given product is only exposed to one or two of them, since the investments are restricted to certain asset classes and certain markets. This observation is shared by [HJL17], "in portfolio risk measurement, we find that a high-dimensional problem may often be decomposed into low-dimensional ones [...] a portfolio loss is typically a summation of losses of individual financial instruments that form the portfolio, while individual financial instruments may depend on only a small number of risk factors." This decomposition means that it is possible to deal with a small number of risk factors for each sub-portfolio, constructing a replicating martingale for each of them. The overall effect of constructing of portfolio of replicating martingales would be additive on the overall dimensionality.

A regress-now method only has to deal with the time-t loss distribution and therefore with a cross-section of the stochastic drivers. In our insurance example, with $d = 5$ and $T = 40$, the time-1 solvency calculation requires dealing with five variables when working with a regress-now method. However, as we show in Section 5.2, there are large quality improvements to be gained by using regress-later methods. This requires working with $dT = 200$ variables instead of only $d = 5$ variables. As mentioned previously, this introduces a combinatorial explosion in basis function when working with full polynomial bases. Whether solving that problem is computationally feasible or not, depends on the computers available. Therefore we can expect *high-dimensional* to be an ever-moving target, similar to *deep* learning, where newer architectures with more layers make older architectures seem not so *deep* anymore. However, no matter how large the computing resources, there will always be a boundary to how large a regression problem can be solved with those resources. One common approach in those cases is to select which variables to work with. Another common approach is to select the most relevant basis functions, adjusting the number of basis functions to a computational budget. At the core of this paper is the idea of finding linear combinations of input variables to then build a full basis on those new input dimensions. We show how that idea can be applied to both polynomials and neural networks, in a regress-later setting. As time goes by and larger models become more common, the example presented in this section—with $T = 40$ and $d = 5$—will be far from considered *high-dimensional*. However, the principle will still remain, that linear dimensionality reduction makes working with high-dimensional models feasible, in those cases where the solution can be found in a subspace of the original input space.

As a final note on the use of linear dimensionality reduction, is it interesting to note that while motivated by the combinatorial explosion of basis functions brought by the regress-later approach,

the linear dimensionality reduction step could also be applied in a regress-now model. Even if regress-now models only need to deal with a subset of the total number of variables, that subset could still be too big to solve a regression on a full polynomial basis. While we do not explore that case in this paper, it would be possible to use the linear dimensionality reduction approach to a regress-now model.

5.2 Results

The results for the variable annuity guarantee confirm those of European option case: 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 12 shows that nested Monte Carlo (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 (1,000) and high dimensions ($T = 40$), providing the worst results in that case. This is driven 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 $2,600 + 101 = 2,701$ for $T = 5$, and $20,100 + 101 = 20,201$ for $T = 40$. 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 3,276 elements, see Table 5, which exceeds the 1,000 available samples. The polynomial LDR has a MApE of less than 0.1% due to only containing 286 basis elements.

Table 12: *Insurance liability, comparison of present value MApE (in percentage points)*

Samples	nMC	Full Polynomial basis		LDR	Neural Network	
		Regress-now	Regress-later	Regress-later	Regress-now	Regress-later
T=5						
1,000	0.3	0.2	61.0	<0.1	0.3	0.1
5,000	0.3	0.1	<0.1	<0.1	0.1	<0.1
10,000	0.2	0.1	<0.1	<0.1	0.1	<0.1
50,000	0.1	<0.1	<0.1	<0.1	<0.1	<0.1
T=40						
1,000	0.5	0.5		0.2	0.6	6.1
5,000	0.3	0.2		0.1	0.2	0.3
10,000	0.3	0.2		0.1	0.2	0.4
50,000	0.2	0.1		<0.1	0.1	0.1

In the estimation of the expected shortfall, shown in Table 13 and Figure 3, and the analysis of the mean relative L_1 error, shown in Table 14, we can observe that regress-later methods dominate over regress-now methods and nested Monte Carlo, with better mean absolute error and standard

Table 13: *Insurance liability, comparison of expected shortfall MApE (in percentage points)*

Samples	nMC	Full Polynomial basis Regress-now	Full Polynomial basis Regress-later	LDR Regress-later	Neural Network Regress-now	Neural Network Regress-later
T=5						
1,000	30.7	80.6	613.9	7.9	198.1	2.9
5,000	11.1	18.6	0.5	5.3	48.7	0.5
10,000	9.2	10.3	0.3	5.1	25.8	0.6
50,000	5.7	3.3	0.2	3.7	6.6	0.8
T=40						
1,000	105.4	226.4		22.9	520.7	14.4
5,000	32.7	64.9		5.7	147.3	11.1
10,000	18.7	35.9		5.9	84.4	10.5
50,000	10.5	10.0		7.2	13.2	0.5

Table 14: *Insurance liability, comparison of relative mean L_1 error (in percentage points)*

Samples	Full Polynomial basis Regress-now	Full Polynomial basis Regress-later	LDR Regress-later	Neural Network Regress-now	Neural Network Regress-later
T=5					
1,000	1.6	61.0	0.2	4.7	0.1
5,000	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
T=40					
1,000	3.3		0.8	10.0	6.1
5,000	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

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 is very sensitive to the choice of hyper-parameters. We provide a sensitivity analysis for hyper-parameters in Sections D and 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 1,000 samples to 5,000 samples. This basis has 3,276 elements, see Table 5, which means that when working with 1,000 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 5,000 to 10,000 and 50,000, see Table 13. This behaviour is not present in the relative mean L_1 error, as evidenced in Table 14. 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.
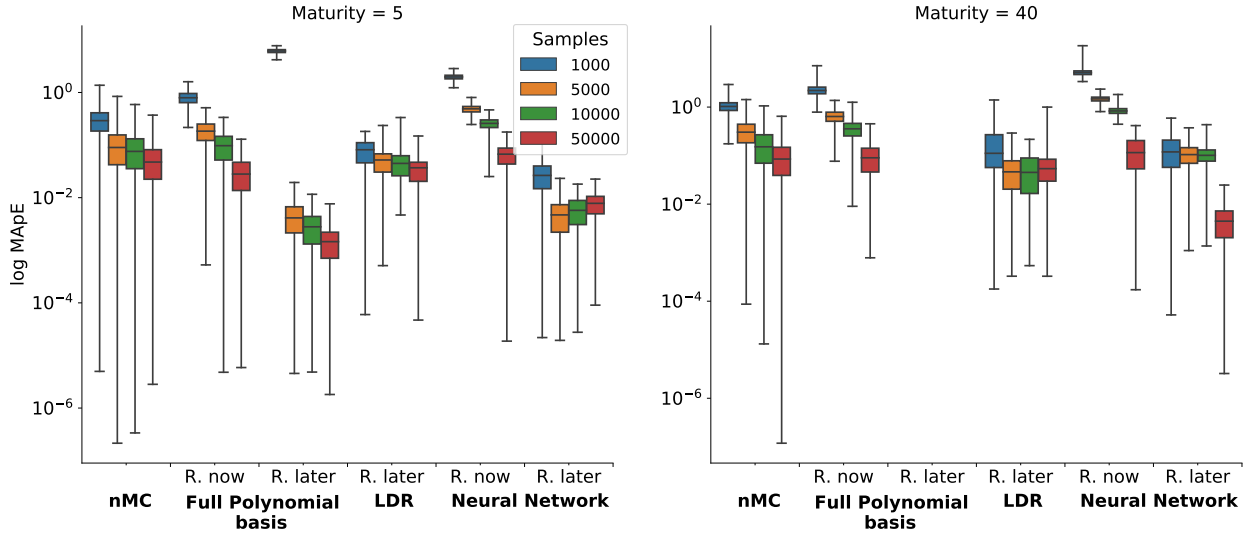
Figure 3: *Distribution of expected shortfall estimates per method for the insurance liability example. Boxes show the upper and lower quartiles of the empirical distributions, while whiskers show their maxima and minima. Due to the logarithmic scale, a visual artifact is introduced by which the inter-quartile 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.*

5.3 Comparison of runtime

The comparison of runtime for the insurance example is very important to determine the relative strength of the methods as feasible solution in the real world. For the reasons described in the introduction, nested Monte Carlo is not a feasible method for this problem, and it is therefore excluded from this comparison.

The full comparison is presented in Table 15. 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, 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 take up to 25 minutes 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 1,000, it takes 33 seconds to solve with $p = 5$ and 279.1 seconds to solve with $p = 10$—the latter is the example shown in Table 15.

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

Table 15: *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					
1,000	0.8	4.3	30.4	4.0	5.5
5,000	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					
1,000	1.3		279.1	4.6	6.3
5,000	2.6		698.9	12.9	15.7
10,000	4.3		1,815.1	22.9	26.5
50,000	17.9		1,472.8	31.7	115.5

6 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 nested Monte Carlo 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.

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 those metrics described above we derive an empirical distribution based on R macro-runs of the 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 [WM05] and [CD14].

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

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

$$\mathrm{MApE\ ES} = \frac{1}{R} \sum_{j=1}^R \frac{|\mathrm{ES}_\alpha[-\Delta \widehat{V}_t^{(j)}] - \mathrm{ES}[-\Delta V_t]|}{\mathrm{ES}[-\Delta V_t]}.$$

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

$$\mathrm{MApE\ PV} = \frac{1}{R} \sum_{j=1}^R \frac{|\widehat{V}_0^{(j)} - V_0|}{V_0}.$$

A.2   Mean relative L_1 error

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

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

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

$$|\mathrm{ES}_\alpha[\hat{V}_t] - \mathrm{ES}_\alpha[V_t]| \leq \frac{1}{\alpha}\mathbb{E}[|\hat{V}_t - V_t|].$$

This shows that for any α, 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 α and only takes into account the distribution beyond the α-th percentile, the mean relative L_1 error takes into account the whole distribution and bounds the expected shortfall error for any α.

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 vec$(X) \sim N(0, I_{dT})$, 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

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

for parameters κ, σ, and function $b(t)$, where W denotes a Brownian motion under the risk-neutral measure, see, e.g., [Gla13]. The nominal price at time t of a zero-coupon bond with maturity T is given by

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

where

$$A(t, T) = \frac{1}{\kappa}(1 - e^{-\kappa(T-t)}),$$

$$C(t, T) = -\kappa h(t, T) + \frac{\sigma^2}{2\kappa^2}\Big[(T - t) + \frac{1}{2\kappa}(1 - e^{-2\kappa(T-t)}) + \frac{2}{\kappa}(e^{-\kappa(T-t)} - 1)\Big]$$

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

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 [Gla13], 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

$$h(t) = h(t, t+1)$$

$$\sigma_Y = \frac{\sigma^2}{\kappa^2}\left(1 + \frac{1}{2\kappa}(1 - e^{-2\kappa}) + \frac{2}{\kappa}(e^{-\kappa} - 1)\right)$$

$$\sigma_{rY} = \frac{\sigma^2}{2\kappa}(1 + e^{-2\kappa} - 2e^{-\kappa})$$

$$\rho_{rY} = \sigma_{rY}/(\sigma_r \sigma_Y)$$

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 Σ 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

$$S_{j,t} = S_{j,t-1} \exp\left(-\sigma_j^2/2 + \sigma_j X'_{j,t}\right),$$

where $j = 3$ for the equity index and $j = 4$ for the real estate index. The equity index EQ_t and the real estate index RE_t are then given by $EQ_t = C_t S_{3,t}$ and $RE_t = C_t S_{4,t}$, 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 [LC92], and they are shown in Table 16.

Table 16: *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)	-6.760120	0.04744
(20, 21, 22, 23, 24)	-6.443340	0.05351
(25, 26, 27, 28, 29)	-6.400620	0.05966
(30, 31, 32, 33, 34)	-6.229090	0.06173
(35, 36, 37, 38, 39)	-5.913250	0.05899
(40, 41, 42, 43, 44)	-5.513230	0.05279
(45, 46, 47, 48, 49)	-5.090240	0.04458
(50, 51, 52, 53, 54)	-4.656800	0.03830
(55, 56, 57, 58, 59)	-4.254970	0.03382
(60, 61, 62, 63, 64)	-3.856080	0.02949
(65, 66, 67, 68, 69)	-3.473130	0.02880
(70, 71, 72, 73, 74)	-3.061170	0.02908
(75, 76, 77, 78, 79)	-2.630230	0.03240
(80, 81, 82, 83, 84)	-2.204980	0.03091
(85, 86, 87, 88, 89)	-1.799600	0.03091
(90, 91, 92, 93, 94)	-1.409363	0.03091
(95, 96, 97, 98, 99)	-1.036550	0.03091
(100, 101, 102, 103, 104)	-0.680350	0.03091
(105, 106, 107, 108)	-0.341050	0.03091

C   Proofs

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

C.1   Proof of Lemma 1

In view of (1) and by orthogonality, we have $\|f-\phi_\theta^\top\beta\|_{L_\mathbb{Q}^2}^2 \geq \|f-\phi_\theta^\top\beta_\theta\|_{L_\mathbb{Q}^2}^2 = \|f\|_{L_\mathbb{Q}^2}^2 - \langle f, \phi_\theta^\top\beta_\theta\rangle_{L_\mathbb{Q}^2}$, for all $(\theta,\beta) \in \Theta \times \mathbb{R}^m$. On the other hand, by assumption (iii) we can write $\beta_\theta = \langle\phi_\theta,\phi_\theta^\top\rangle_{L_\mathbb{Q}^2}^{-1}\langle\phi_\theta, f\rangle_{L_\mathbb{Q}^2}$ and hence $\langle f, \phi_\theta^\top\beta_\theta\rangle_{L_\mathbb{Q}^2} = \langle f, \phi_\theta^\top\rangle_{L_\mathbb{Q}^2}\langle\phi_\theta,\phi_\theta^\top\rangle_{L_\mathbb{Q}^2}^{-1}\langle\phi_\theta, f\rangle_{L_\mathbb{Q}^2}$. Hence (5) is equivalent to (1) and

$$\max_{\theta\in\Theta}\langle f, \phi_\theta^\top\rangle_{L_\mathbb{Q}^2}\langle\phi_\theta,\phi_\theta^\top\rangle_{L_\mathbb{Q}^2}^{-1}\langle\phi_\theta, f\rangle_{L_\mathbb{Q}^2}.$$

By the assumptions of the lemma, $\theta \mapsto \langle f, \phi_\theta^\top \rangle_{L_\mathbb{Q}^2} \langle \phi_\theta, \phi_\theta^\top \rangle_{L_\mathbb{Q}^2}^{-1} \langle \phi_\theta, f \rangle_{L_\mathbb{Q}^2}$ is continuous, and hence attains its maximum on the compact set Θ. This completes the proof.

C.2   Proof of Theorem 1

Theorem 1 follows from Lemmas 2 and 3 below.

Lemma 2. *Let $A, \tilde{A} \in \mathbb{R}^{dT \times p}$ with full rank and $b, \tilde{b} \in \mathbb{R}^p$. The following are equivalent:*

(i) $\ker A^\top = \ker \tilde{A}^\top$

(ii) $\tilde{A} = AS$, *for some invertible $p \times p$-matrix S*

(iii) $\mathrm{span}\{\phi_{(A,b),1}, \ldots, \phi_{(A,b),m}\} = \mathrm{span}\{\phi_{(\tilde{A},\tilde{b}),1}, \ldots, \phi_{(\tilde{A},\tilde{b}),m}\}$

Proof. (i)$\Leftrightarrow$(ii): is elementary.

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

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

Lemma 3. *Let $A, \tilde{A} \in \mathbb{R}^{dT \times p}$ with full rank. The following are equivalent:*

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

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

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

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

C.3   Proof of Theorem 2

As $V_p(\mathbb{R}^{dT})$ is a compact manifold, it follows by inspection that the assumptions of Lemma 1 are met. Hence there exists a minimizer of (5). The non-uniqueness statement is proved by means of the following counterexample. Assume $f(x) = f(Vx)$ and the pushforward $V_* \mathbb{Q} = \mathbb{Q}$ for some orthogonal $dT \times dT$-matrix V. Then, for any $A \in V_p(\mathbb{R}^{dT})$ and $\beta \in \mathbb{R}^m$, we have $\|f - \phi_A^\top \beta\|_{L_\mathbb{Q}^2} = \|f - \phi_{V^\top A}^\top \beta\|_{L_\mathbb{Q}^2}$. But $\mathrm{span}\{\phi_{A,1}, \ldots, \phi_{A,m}\} \neq \mathrm{span}\{\phi_{V^\top A,1}, \ldots, \phi_{V^\top A,m}\}$ in general by Theorem 1. This completes the proof of Theorem 2.

C.4 Proof of Theorem 3

We follow the heuristic arguments of [HLXZ20].[5] 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 [Sch12, Theorem 3.1.2]. That is, its directional derivative $\nabla_v h(x) = \lim_{\epsilon \downarrow 0} (h(x + \epsilon v) - h(x))/\epsilon$ exists for all $x, v \in \mathbb{R}^{dT}$, and it provides a first order approximation, $\lim_{y \to x} \|h(y) - h(x) - \nabla_{y-x} h(x)\| / \|y - x\| = 0$. 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 [Sch12, 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

$$
D_{\nabla_v \phi_{(a_i, b_i)}} = \begin{cases} \emptyset, & \text{if } v^\top a_i = 0, \\ H_{(a_i, b_i)}, & \text{otherwise,} \end{cases}
$$

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

$$
H_{(a_i, b_i)} = D_{\nabla_v \phi_{(a_i, b_i)}} = D_{\sum_{j=1}^m c_j \nabla_v \phi_{(\tilde{a}_j, \tilde{b}_j)}} \subseteq \cup_{j=1}^m D_{\nabla_v \phi_{(\tilde{a}_j, \tilde{b}_j)}} \subseteq \cup_{j=1}^m H_{(\tilde{a}_j, \tilde{b}_j)},
$$

where we used the obvious relation $D_{t_1 h_1 + t_2 h_2} \subseteq D_{h_1} \cup D_{h_2}$, for functions h_1, h_2 and real coefficients t_1, t_2. This implies that $(a_i, b_i) \in \{\pm(\tilde{a}_1, \tilde{b}_1), \ldots, \pm(\tilde{a}_m, \tilde{b}_m)\}$. Since i was arbitrary, we obtain $\{\pm(a_1, b_1), \ldots, \pm(a_m, b_m)\} \subseteq \{\pm(\tilde{a}_1, \tilde{b}_1), \ldots, \pm(\tilde{a}_m, \tilde{b}_m)\}$. A similar argument for $(\tilde{a}_i, \tilde{b}_i)$ in lieu of (a_i, b_i) implies the converse inclusion. This proves (i).

For the proof of (ii) we argue by contradiction. Suppose (13) does not hold, so that $\phi_{(a_i, b_i)} \in \text{span}\{\phi_{(a_j, b_j)} \mid j \neq i\}$ for some i. Hence (12) holds for $(\tilde{a}_j, \tilde{b}_j) := (a_j, b_j)$ for all $j \neq i$, and $(\tilde{a}_i, \tilde{b}_i) := (a_j, b_j)$ 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 3.

Remark 1. *One may reckon that* (12) *and* (13) *together imply*

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

However, this is not true in general. Indeed, let $(a_1, b_1), \ldots, (a_3, b_3) \in S_{dT}$ *be as in Example 4, 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* $\tilde{F} = \{\phi_{(a,b)} \mid (a, b) \in \tilde{S}\}$ *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* $\text{span } F = \text{span } \tilde{F}$. *But* $S \neq \tilde{S}$.

---

[5] We also complete some arguments in [HLXZ20], who do not explain what kind of derivative "$\nabla \sum_{i=1}^m c_j \phi_{(a_j, b_j)}$" stands for.

C.5 Proof of Theorem 4

We prove the theorem by means of two counterexamples. First, let $d = T = 1$, $\mathbb{Q} = N(0,1)$, 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 $\theta_n = ((a_n, b_n), (0,1)) \in (S_1)^2$. Then $\theta_n \to ((0,1),(0,1))$ as $n \to \infty$. On the other hand, for $\beta_n = (1/b_n, -a_n/b_n)^\top$, we have

$$\phi_{\theta_n}(x)^\top \beta_n = \frac{1}{b_n}(a_n x + b_n)^+ - \frac{a_n}{b_n} x^+ \to 1_{[0,\infty)}(x) \quad \text{in } L_{\mathbb{Q}}^2 \text{ as } n \to \infty.$$

Hence $\inf_{(\theta,\beta) \in (S_1)^2 \times \mathbb{R}^2} \|f - \phi_\theta^\top \beta\|_{L_{\mathbb{Q}}^2} = \lim_n \|f - \phi_{\theta_n}^\top \beta_n\|_{L_{\mathbb{Q}}^2} = 0$, but the infimum is not attained, $\|f - \phi_\theta^\top \beta\|_{L_{\mathbb{Q}}^2} > 0$ for all $(\theta, \beta) \in (S_1)^2 \times \mathbb{R}^2$. This proves the non-existence statement.

For the non-uniqueness, 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 $\|f - \phi_{(A,b)}^\top \beta\|_{L_{\mathbb{Q}}^2} = \|f - \phi_{(V^\top A, b)}^\top \beta\|_{L_{\mathbb{Q}}^2}$. But $\text{span}\{\phi_{(A,b),1}, \ldots, \phi_{(A,b),m}\} \neq \text{span}\{\phi_{(V^\top A,b),1}, \ldots, \phi_{(V^\top A,b),m}\}$ in general by Theorem 3. This completes the proof of Theorem 4.

D Sensitivity of polynomial LDR to hyper-parameters

The polynomial LDR method has two hyper-parameters, the target dimensionality p and the polynomial degree δ. 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 linear dimensionality reduction 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.

D.1 Starting point A_0

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

A 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 dTp random samples from $N(0,1)$ and arrange them into an $dT \times p$ matrix B. Then $A_0 = B(B^\top B)^{-1/2}$ 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 $(i = j \wedge i \neq p) \wedge B_{ip} = \frac{1}{\sqrt{dT-p+1}}$ if $i \geqslant p$. 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 $p = 3$:

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

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 3.4:

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

Since this method, which we call *folding*, provides the best results we also use it in Sections 4 and 5. In the latter we work with $T = 5$, $d = 5$ and $p = 10$, and leads to a starting point (in block notation):

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

In Tables 17 and 18 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—financial derivatives and insurance liabilities—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 17: *Insurance liability, comparison of expected shortfall MApE (in percentage points) for different starting points in polynomial LDR basis*

Samples	Folding	Diagonal	Random
T=5			
1,000	7.9	14.9	16.6
5,000	5.3	17.5	31.5
10,000	5.1	23.6	31.8
50,000	3.7	31.0	32.4
T=40			
1,000	22.9	38.0	63.9
5,000	5.7	54.7	69.4
10,000	5.9	53.0	68.2
50,000	7.2	49.2	67.0

Table 18: *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			
1,000	0.2	1.0	1.2
5,000	0.2	0.8	1.0
10,000	0.2	0.9	1.0
50,000	0.2	0.9	1.0
T=40			
1,000	0.8	1.7	1.7
5,000	0.3	1.3	1.5
10,000	0.4	1.3	1.5
50,000	0.4	1.2	1.5

D.2 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 19 and 20 confirm the expected effect of changing this parameter: larger values of p produce better results (since $f^\star$ 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 1,000 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—5,000 and above—the error goes down as expected.

Table 19: *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				
1,000	22.3	14.9	228.0	767.9
5,000	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 20: *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				
1,000	1.0	1.0	3.1	37.3
5,000	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

E   Sensitivity of neural network to hyper-parameters

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 5, 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 hyperparameters.

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 function $f^\star$.

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 5. The results are summarized in Tables 21 and 22. 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.

Table 21: *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				
1,000	2.9	2.2	3.1	3.9
5,000	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				
1,000	14.4	14.5	17.5	14.1
5,000	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

Table 22: *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				
1,000	0.1	0.1	0.1	0.1
5,000	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				
1,000	6.1	6.4	2.8	6.5
5,000	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

The first alternative is to use the theoretical minimum width for the network, as described in [HS17]. 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 this is valid for neural networks of arbitrary length, 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 2,231 parameters. This can be matched by using a neural network with $p = 18$ and $p = 11$ nodes respectively. This alternative provide 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.

References

[ABG07]   P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007. 19

[AC03]   Giulia Andreatta and Stefano Corradin. Valuing the surrender options embedded in a portfolio of italian life guaranteed participating policies: a least squares monte carlo approach. In *Proceedings of "Real option theory meets practice", 8th Annual International Conference, Montreal*, 2003. 3

[ADGA08]   Kevin C Ahlgrim, Stephen P DArcy, Richard W Gorvett, and FRM ARM. A comparison of actuarial financial scenario generators. *Variance*, 2(1):111–134, 2008. 26

[BDM15]   Mark Broadie, Yiping Du, and Ciamac C Moallemi. Risk estimation via regression. *Operations Research*, 63(5):1077–1097, 2015. 4, 5, 9

[BF19]   Lotfi Boudabsa and Damir Filipović. Machine learning with kernels for portfolio valuation and risk management. Swiss Finance Institute Research Paper No. 19-34, 2019. https://ssrn.com/abstract=3401539. 17

[Car96]   Jacques F Carriere. Valuation of the early-exercise price for options using simulations and nonparametric regression. *Insurance: mathematics and Economics*, 19(1):19–30, 1996. 3

[CD14]   Tianfeng Chai and Roland R Draxler. Root mean square error (rmse) or mean absolute error (mae)?–arguments against avoiding rmse in the literature. *Geoscientific model development*, 7(3):1247–1250, 2014. 31

[CFM+18]   Gilberto Castellani, Ugo Fiore, Zelda Marino, Luca Passalacqua, Francesca Perla, Salvatore Scognamiglio, and Paolo Zanetti. An investigation of machine learning approaches in the solvency ii valuation framework. *Available at SSRN 3303296*, 2018. 4, 5

[CG15]   John P Cunningham and Zoubin Ghahramani. Linear dimensionality reduction: Survey, insights, and generalizations. *The Journal of Machine Learning Research*, 16(1):2859–2900, 2015. 14

[DFS03]   D. Duffie, D. Filipović, and W. Schachermayer. Affine processes and applications in finance. *Ann. Appl. Probab.*, 13(3):984–1053, 2003. 17

[DS06]   Freddy Delbaen and Walter Schachermayer. *The mathematics of arbitrage*. Springer Finance. Springer-Verlag, Berlin, 2006. 16

[Duo19]   Quang Dien Duong. Application of bayesian penalized spline regression for internal modeling in life insurance. *European Actuarial Journal*, 9(1):67–107, 2019. 4, 5

[FA21]   Lucio Fernandez-Arjona. A neural network model for solvency calculations in life insurance. *Annals of Actuarial Science*, page to appear, 2021. 16

[FAF20]   Lucio Fernandez-Arjona and Damir Filipović. Benchmark and training data for replicating financial and insurance examples, 2020. Zenodo. 3

[FS04]   Hans Föllmer and Alexander Schied. *Stochastic finance*, volume 27 of *De Gruyter Studies in Mathematics*. Walter de Gruyter & Co., Berlin, extended edition, 2004. An introduction in discrete time. 7

[GJ10]   Michael B Gordy and Sandeep Juneja. Nested simulation in portfolio risk measurement. *Management Science*, 56(10):1833–1848, 2010. 5

[Gla13]   Paul Glasserman. *Monte Carlo methods in financial engineering*, volume 53. Springer Science & Business Media, 2013. 32

[GY02]   Paul Glasserman and Bin Yu. Simulation for american options: Regression now or regression later? In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, pages 213–226. Springer, 2002. 3, 10

[HB20]   Hongjun Ha and Daniel Bauer. A least-squares monte carlo approach to the estimation of enterprise risk. https://danielbaueracademic.files.wordpress.com/2020/04/lsm_habauer2020.pdf, 2020. 4, 5

[HC18]   Jeffrey M Hokanson and Paul G Constantine. Data-driven polynomial ridge approximation using variable projection. *SIAM Journal on Scientific Computing*, 40(3):A1566–A1589, 2018. 5, 19

[HGA15]   Wen Huang, Kyle A Gallivan, and P-A Absil. A broyden class of quasi-newton methods for riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685, 2015. 17, 19

[HJL17]   L Jeff Hong, Sandeep Juneja, and Guangwu Liu. Kernel smoothing for nested estimation with application to portfolio risk measurement. *Operations Research*, 65(3):657–673, 2017. 4, 5, 26

[HLXZ20]   Juncai He, Lin Li, Jinchao Xu, and Chunyue Zheng. Relu deep neural networks and linear finite elements. *Journal of Computational Mathematics*, 38(3):502–527, 2020. 36

[HS17]   Boris Hanin and Mark Sellke. Approximating continuous functions by relu nets of minimal width. *arXiv preprint arXiv:1710.11278*, 2017. 41

[Kan08]   Raymond Kan. From moments of sum to moments of product. *Journal of Multivariate Analysis*, 99(3):542 – 554, 2008. 14, 15

[LBBH98]   Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 5

[LC92]   Ronald D. Lee and Lawrence R. Carter. Modeling and forecasting u. s. mortality. *Journal of the American Statistical Association*, 87(419):659–671, 1992. 23, 33

[LG03]   Shing-Hoi Lee and Peter W Glynn. 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, 2003. 5

[LS01]   Francis A. Longstaff and Eduardo S. Schwartz. Valuing american options by simulation: A simple least-squares approach. *The Review of Financial Studies*, 14(1):113–147, 2001. 3

[MM94]   Dilip B Madan and Frank Milne. Contingent claims valued and hedged by pricing and investing in a basis. *Mathematical Finance*, 4(3):223–245, 1994. 3

[PS16]   Antoon Pelsser and Janina Schweizer. The difference between LSMC and replicating portfolio in insurance liability modeling. *European actuarial journal*, 6(2):441–494, 2016. 3, 4, 5

[PVG+11]   F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 17, 20

[RL18]   Jimmy Risk and Michael Ludkovski. Sequential design and spatial modeling for portfolio tail risk measurement. *SIAM Journal on Financial Mathematics*, 9(4):1137–1174, 2018. 4, 5

[Sch12]   Stefan Scholtes. *Introduction to piecewise differentiable equations*. SpringerBriefs in Optimization. Springer, New York, 2012. 36

[Sul15]   T. J. Sullivan. *Introduction to uncertainty quantification*, volume 63 of *Texts in Applied Mathematics*. Springer, Cham, 2015. 13

[Tib96]   Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996. 20

[TKW16]   James Townsend, Niklas Koep, and Sebastian Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759, 2016. 19

[WM05]   Cort J Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. *Climate research*, 30(1):79–82, 2005. 31

[ZHT+07]   Hui Zou, Trevor Hastie, Robert Tibshirani, et al. On the degrees of freedom of the lasso. *The Annals of Statistics*, 35(5):2173–2192, 2007. 22

Swiss Finance Institute

Swiss Finance Institute (SFI) is the national center for fundamental research, doctoral training, knowledge exchange, and continuing education in the fields of banking and finance. SFI's mission is to grow knowledge capital for the Swiss financial marketplace. Created in 2006 as a public–private partnership, SFI is a common initiative of the Swiss finance industry, leading Swiss universities, and the Swiss Confederation.

swiss:finance:institute