Machine learning applications in the insurance industry

Dissertation
submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wirtschaftswissenschaften, Dr. oec.
(corresponds to Doctor of Philosophy, PhD)

presented by

Lucio Fernandez-Arjona Berchansky
from Argentina and Spain

approved in September 2021 at the request of

Prof. Dr. Karl Schmedders
Prof. Dr. Damir Filipovic
Prof. Dr. Erich Walter Farkas

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, 22.09.2021

The Chairman of the Doctoral Board: Prof. Dr. Steven Ongena

Dedication

To my wife, Mariana. Without your love and support none of this would have been possible.

To Tomás and Camila, who do not care about dissertations but will enjoy more time with their dad from now on.

Acknowledgments

My deepest gratitude goes to my supervisor, Karl Schmedders, for having accepted me as an external candidate, for his guidance, and continuous support during these years. This journey would not have even started without his encouragement. To my co-author Damir Filipovic, I want to deeply thank for his patience and for all that I learned from him while working together. His insights improved our work in countless ways. A big thank you also goes to Walter Farkas, for agreeing to be part of my dissertation committee. I thank my colleagues at the Chair of Quantitative Business Administration for our discussions and their valuable inputs.

Contents

General Introduction

For those working in any field where data and statistics play a large role, machine learning has become a term that is often used and sometimes misunderstood. Above all, it has become a tool of undeniable usefulness when applied to the right problem.

The last decade has been called the *AI spring*, a reference made to contrast with the previous *AI winter*—a period of reduced funding and interest in artificial intelligence research during the 1990s. The fact that these days it is more common to hear about machine learning—a sub-field—than artificial intelligence is a reflection of the bad reputation that artificial intelligence had for many years, after a hype cycle was followed by disappointment and criticism.

The last ten years have seen a string of successes for the field of machine learning: IBM Watson's Jeopardy! victory, deep learning algorithms exceeding the human performance at the ImageNet challenge, AlphaGo defeating world Go champion Lee Sedol, and many others. These are the product—in our opinion—of the convergence of existing algorithms, the availability of cheap and powerful computing resources, and the relatively easy access to large datasets.

The result has been an explosion in both academic research and commercial applications. The Scopus database registered 526 papers with the words *machine learning* in the year 2000, 4,543 papers in the year 2010 and 60,237 papers in the year 2020. The interest is not limited to academia: the venture capital database CB Insights reports that funding for AI startups increased from USD 4.3 billions in 2014 to USD 26.6 billions in 2019. At the same time, Linkedin—the social network for professional networking— reports a 74% annual growth on published job searches in the previous four years for artificial intelligence specialists (Linkedin (2020)).

As part of an industry that relies extensively on data and predictive models for business decisions, insurance companies can make use of machine learning models in their operations. Having easy access to the main algorithms in use today—courtesy of open source software—and easy access to computational resources—provided by cloud service providers—it should be *only* a matter of curating datasets that are large enough to feed

those algorithms. This is, however, not a small challenge.

Digital-native industries—such as online advertising—can build large datasets out of the interactions between customers and their own web pages with no larger marginal cost than that of storage. Industries that grew up in an analog world, where many processes are manual and the information is more limited, cannot so easily build the large datasets required for machine learning applications.

Despite the difficulties, the potential of applying machine learning in insurance is too large to ignore. In pricing applications, the well-known adverse selection problem prevents companies from being overly complacent in a competitive market. If other insurance companies successfully used machine learning to identify the better risks—those policyholders with lower probability of filing a claim—they could lower their prices, attract the best customers, and avoid the worst ones.

In this dissertation we explore one possible use of machine learning in the insurance industry. We do not look into applications to pricing—which many others have explored and keep actively working on. Instead, we look into an application for risk management.

Risk management models have evolved greatly in the last ten to fifteen years, driven—among other reasons—by regulation that arose out of the 2008 banking crisis. More than ten years later, these models are still evolving. The natural and unavoidable low frequency of tail events is one of the biggest problems they face. The computational difficulties when working with large models and high-dimensional distributions is another one. This dissertation aims to make a contribution in this latter problem.

Paper 1 describes the current practice regarding market risk modelling for a life insurance company. Unlike banks—which for the most part only need to focus on the market risk of their assets—life insurance companies must model both assets and liabilities to understand the market risk exposure in their balance sheets.

The modelling of insurance liabilities starts with a projection of future cash flows under many different market risk scenarios. This projection has a long horizon, because life insurance policies can be active over many decades. If a risk-neutral distribution is chosen to generate future cash flows, then it is possible to calculate the Monte Carlo

price of the insurance liability. This is usually necessary because there are no closed-form formulas for the price of most life insurance products. Using a Monte Carlo approach means that thousands of cash flow paths must be generated.

The process described above is very common in life insurance and it does not represent a major challenge in itself. However, the challenges begin when the Monte Carlo pricing of the liabilities is combined with the Monte Carlo evaluation of a risk metric. This is called a nested Monte Carlo approach. When a risk metric needs to be approximated using Monte Carlo simulations, and each of those simulations requires in turn a calculation that is approximated using Monte Carlo simulations, the number of simulations grows extremely quickly and the total computational cost can easily exceed what is practical.

This first paper describes one of the most common approaches used by insurance companies to reduce the computational cost of solvency calculations: the replicating portfolios approach. This method is based on the concept of static replication of derivatives. Given a number of simulated paths of the cash flows of the insurance liability, the method finds the portfolio of financial instruments that minimizes some distance function between the path of cash flows of the liabilities and the portfolio.

Once the optimal portfolio has been approximated as well as possible from a finite sample, and assuming that there are closed-form pricing formulas for all financial instruments, the inner Monte Carlo approach—the one related to pricing—can be replaced with those formulas, therefore greatly reducing the time it takes to complete all calculations.

The reduction in computational cost is not just a matter of efficiency, or reduction of costs. The difference is so large that without a method like replicating portfolios insurance companies would not be able to satisfy the regulatory requirements to annually or quarterly report their solvency position.

Replicating portfolios is not the only approach used by insurance companies. There are several popular *proxy* models—known as *surrogate* models, or *response surface* models in the field of engineering—that are used to mimic the behavior of the simulation model as closely as possible while being computationally cheaper to evaluate. This paper only focuses on replicating portfolios, but the following papers describe some alternatives in

more detail.

Paper 1 describes how one European insurance company uses a linear programming approach to find the optimal replicating portfolio, and discusses all the practical issues associated with this type of proxy modelling. From the choice of simulations—where a type of stratified sampling is used—to the choice of financial instruments, there are many details to the construction of a practical implementation of the method of replicating portfolios. Most importantly, the paper discusses at length the quality metrics that the company used to verify the validity of the solutions.

The paper also describes the iterative process for generating a replicating portfolio, which requires the user to produce several candidates. These candidates differ—among other things—in the subset of the asset universe used as input to the optimization. That means that the the process is not entirely automated, and the key step of feature selection—as it is called in machine learning—is left for the user to do by analysing the quality metrics of the competing candidate portfolios.

Paper 2 introduces the idea of using a neural network as a proxy model, instead of using a portfolio of financial instruments as described in Paper 1. The paper's first contribution is the introduction of a reproducible replicating portfolio approach. As opposed to the replicating portfolio approach described in Paper 1—which requires manual input and a user's intuition to set the different hyperparameters—this approach is suitable for benchmarking because it consistently produces the same results when provided the same inputs.

The automated replicating portfolio approach used for benchmarking in this paper replaces the manual feature selection of the industry approaches with a LASSO regressor. The LASSO regressor is very effective at generating sparse solutions, eliminating the need to feed subsets of the asset universe as done during the manual production of replicating portfolios. The LASSO regressor has only one parameter: the regularization parameter. The paper uses the Akaike information criterion to automatically select the best regularization parameter. The combination of LASSO regressor and the Akaike information criterion produces a model that does not require manual steps for feature selection, while

remaining robust to overfitting.

The second contribution of this paper is a risk-neutral valuation formula for single-layer neural networks with a ReLu activation function. This is necessary to efficiently replace the inner Monte Carlo simulations of the nested Monte Carlo approach with a formula of comparable efficiency as the closed form pricing formulas of the replicating portfolios.

The third and central contribution of this paper is the description of a proxy modelling technique based on neural networks for the problem of the risk-neutral valuation of the life insurance liabilities in the context of solvency calculations. This neural network approach is compared to the replicating portfolio method in both their qualitative and quantitative dimensions. The latter comparison is based on comparing the estimation error in the solvency capital of a variable annuity guarantee using nested Monte Carlo, replicating portfolios, and neural networks.

Paper 3 also looks into the application of machine learning to solvency calculations. Compared to the previous papers, it provides a more detailed look at the mathematical framing of the problem and, on the practical side, much more comprehensive quantitative comparisons across methods.

This paper frames the problem of building a proxy model as that of replicating a martingale, and presents a framework for understanding and comparing some key differences across several proxy methods: the functional basis, the learning of features—or lack thereof—and whether it belongs to the family of regress-now or regress-later methods.

Unlike the previous two papers, this paper does not look at replicating portfolios and instead focuses on polynomial methods. Within that family, Least Squares Monte Carlo (LSMC) is the best known in the insurance industry and it is very often used for solvency calculations. Across the dimensions mentioned before this method is classified as polynomial basis method, without feature learning and of the regress-now family.

The paper explores a method—which we refer to as polynomial-based replicating martingale—that uses a polynomial basis in a regress-later setting, and which adds feature learning to achieve a linear dimensionality reduction in the inputs. This dimensionality reduction allows the model to deal with high-dimensional inputs that other polynomial, regress-later

models would not be able to work with.

The paper presents extensive comparisons between three representatives of the polynomial family of methods. The LSMC method—which is a regress-now without feature learning—, a polynomial regress-later method without feature learning, and the full replicating martingale, which is a regress-later method with feature learning. These comparisons are based on a European call and a fictitious insurance contract which includes an investment account and a path-dependent guarantee similar to those included in variable annuity contracts. The comparisons explore the quality of the expected shortfall estimates under a range of sample sizes and contract maturities.

The comparison across a range of sample sizes allows us to explore the problem of overfitting that naturally arises when the number of samples is low relative to the number of parameters in the model. The use of linear dimensionality reduction helps to reduce the number of parameters and enables fitting larger dimensional models with fewer samples. The comparison across contract maturities allows us to explore the change in the quality of the estimations under a change in the dimensionality of the inputs.

Establishing a link with Paper 2, Paper 3 also compares a neural network basis, under both regress-now and regress-later. In the quality comparisons the latter shows the best results of the group, including the comparison against polynomial basis methods.

Unlike Papers 1 and 2—which focus on the calculations and practical outcomes of the methods—Paper 3 includes a section of theoretical results—such as upper bounds on errors, and the conditions for the existence and uniqueness of a solution to the optimization problem that arises from feature learning. We find that the solution is in general not unique.

In summary, the dissertation explores one area of applications of machine learning in the insurance industry. Using non-linear optimization to allow a data-driven construction of the necessary features—instead of requiring users to manually select those out of a predetermined set—allows fully automated processes for proxy model building. This is an example of one machine learning—imitating—another machine's behaviour, which is

only known from black-box observations. The applications of these ideas can go beyond solvency calculations and extend to other situations where there are black-box systems involved. While it is very likely that there is a component of excessive promotion—hype– of the benefits of machine learning, it is undeniable that it enables many applications which were formerly out of reach. In the example of solvency calculations studied in this dissertation, the benefits of automated systems based on non-linear optimization are clear: higher quality approximations and a reduction of manual work.

Machine learning applications in the insurance industry

1. A large-scale optimization model for replicating portfolios in the life insurance industry.

   This paper is published as Adelmann, Maximilian, Lucio Fernandez-Arjona, Janos Mayer, and Karl Schmedders. "A large-scale optimization model for replicating portfolios in the life insurance industry.", Operations Research, Volume 69 (Number 4), 1134-1157, 2021.

2. A neural network model for solvency calculations in life insurance

   This paper is published as Fernandez-Arjona, Lucio. "A neural network model for solvency calculations in life insurance." Annals of Actuarial Science, Volume 15 (Number 2), 259-275, 2021.

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

   A later revision of this paper has been approved for publication in Mathematical Finance.

1 A large-scale optimization model for replicating portfolios in the life insurance industry

Maximilian Adelmann, University of Zurich

Janos Mayer, University of Zurich

Lucio Fernandez-Arjona, University of Zurich

Karl Schmedders, IMD

Abstract

Replicating portfolios have emerged as an important tool in the life insurance industry, used for the valuation of companies' liabilities. This paper describes the replicating portfolio (RP) model for approximating life insurance liabilities in place in a large global insurance company. We describe the challenges presented by the latest solvency regimes in Europe and how the RP model enables this company to comply with the Swiss Solvency Test. The model minimizes the L_1 error between the discounted life insurance liability cash flows and the discounted RP cash flows over a multi-period time horizon for a broad range of different future economic scenarios. A numerical application of the RP model to empirical data sets demonstrates that the model delivers RPs that match the liabilities and perform well for economic capital calculations.

We thank Regine Scheder for helpful discussions on the subject and comments on earlier implementations of the model. We are also indebted to seminar participants at the University of Zurich, the Hoover Institution at Stanford, the EURO2015 in Glasgow, and the Young Finance Scholar Conference at the University of Sussex for helpful comments. We thank Dave Brooks for excellent editorial help. Furthermore, we would like to express our gratitude to two anonymous referees, an anonymous associate editor, and the area editor Andrés Weintraub for thoughtful reviews of an earlier version.

1.1 Introduction

During the last decade, replicating portfolios have emerged as an important tool in the life insurance industry for valuing companies' liabilities. In this paper we describe the replicating portfolio (RP) model for approximating life insurance liabilities in place in a large global insurance company. We describe the challenges presented by the latest solvency regimes in Europe and how the RP model enables this company to comply with the Swiss Solvency Test. The model minimizes the L_1 error between the discounted life insurance liability cash flows and the discounted RP cash flows over a multi-period time horizon for a broad range of different future economic scenarios. A numerical application of the RP model to empirical data sets demonstrates that the model delivers RPs that match the liabilities and perform well for economic capital calculations. Importantly, we also discuss the critical role played by the computed RPs in the risk management process of the insurance company in question.

Life insurance contracts can be described in terms of their future cash flows. This property allows them to be valued using a discounted cash flow approach. Some of these contracts have cash flows that depend on financial markets' variables. Such a contract is, therefore, a financial derivative. In arbitrage-free pricing, the price of a derivative can be calculated as the expected value of its cash flows under the risk-neutral measure. For many derivatives, this expected value leads to a closed-form formula. This is not true in general for life insurance liability models, even for relatively simple contracts. Calculating the market-consistent value of these liabilities requires calculating a complex integral and the most common approach used is Monte Carlo integration. This approach leads to long computing times because the cash flows must be calculated for thousands or even millions of contracts over an entire horizon of 40 or more years. For example, calculating the market-consistent value of a large book of business with complex guarantees can take from a few minutes to a few hours depending on the model's detailedness and the computing resources available. This entire process must be repeated for every change in an economic condition, such as, for example, an interest rate increase of 1 percent. Performing these calculations for ten or twenty changes in economic conditions can take

hours or days.

Under solvency regimes that require the calculation of economic capital, insurance companies must perform a valuation of their liabilities not dozens but thousands of times. Examples of regulatory solvency frameworks with such requirements regarding internal models include Solvency II and the Swiss Solvency Test. Insurance companies are unable to run the required calculations directly. Instead they must find methods that can approximate the Monte Carlo valuation accurately in seconds instead of hours. In this paper, we present such a method, as implemented by Zurich Insurance Group Ltd., one of the largest insurance companies in the world. We describe how the model uses cash flow matching to calculate a portfolio of financial derivatives that replicates the behavior of those liabilities with the accuracy necessary to enable the calculation of economic capital. We also present a quality measurement formula that can be used to verify that the RP model is suited to the task of representing liabilities in solvency calculations.

This paper is not the first reference in the literature to an RP model, but is the first to fully describe a model that has been implemented in real-world conditions. The model has a significant impact on Zurich's business process and is essential for economic capital calculation. Furthermore, we present quality criteria that link the results of the model directly to the calculation of economic capital, unlike other commonly used metrics—such as the R^2 for example—which are generic and cannot provide an interpretation in terms of capital requirements.

In a detailed, real-life, numerical example we present the implementation of our model. We describe the scenario generation process, the liability structure, and the function of the candidate instruments that are used to calculate the RP. We also provide an example of the RP selection process—based on certain quality criteria—and an interpretation of the optimal portfolio in relation to the liabilities. Finally, we discuss how the model fits into Zurich's risk management framework. RPs contribute to the understanding of Zurich's

The Zurich Insurance Group Ltd. ("Zurich") is a global insurance company headquartered in Zurich, Switzerland. It employs approximately 54,000 people worldwide and operates in more than 210 countries and territories. It is a Swiss blue chip with a market value of around USD 45 billion; see `"www.zurich.com/en/about-us/facts-and-figures"` (last accessed August 18, 2017) for further information.

liabilities and they are essential to Zurich's efforts to meet regulatory requirements under the Swiss Solvency Test.

The Literature

Schrager (2008) describes RPs and emphasizes that they are an important tool and can be used for risk-based solvency calculations. Koursaris (2011) provides an introduction to RPs. Boekel et al. (2009) demonstrate the use of RPs as a pool of instruments to reproduce the cash flows of a pool of liabilities across a large number of stochastic scenarios. They point out such portfolios' great benefits for the recalculation of the effects of financial market developments and as a sophisticated tool for risk aggregation. Daul and Gutiérrez Vidal (2009) present various techniques for constructing an RP based on the L_2 norm for cash flow matching. They analyze different approaches and study practical aspects using case studies indicating the robustness of the replication technique and presenting an effective way of evaluating the quality of the replication. Seemann (2009) explains the theory of RPs and the complete RP process at great length, and provides a detailed numerical example. For the RP calibration process he suggests the L_2 norm. He points out the importance of the available set of candidate instruments and presents an outlook for the use of RPs in the life insurance industry. Devineau and Chauvigny (2011) present common applications of the RP technique. They urge the practitioner to consider several factors for an optimization process, including the makeup of the portfolio, the parameters of its instruments, the selection of the optimization program, and the type of variable that has to be replicated. They provide different calibration methods, which are all based on the L_2 norm. Oechslin et al. (2007) propose a terminal-value matching approach for RPs as a superior alternative to cash flow matching models. Natolski and Werner (2014a) compare different quadratic measures for RPs and introduce a new discounted terminal-value matching approach. They do not allow for constraints in the optimization problem. Burmeister and Mausser (2009) stress the advantages of an

Replicating portfolios have a long history in derivative pricing. Since this literature is of little if any relevance to the application of RPs in the insurance industry, we do not review this vast literature and just refer to Fouque et al. (2000), who provide an introduction to RPs in the context of the Black–Scholes model for derivative pricing. In this context RPs are used in dynamic hedging strategies.

RP with a small number of instruments. They establish trading constraints in their optimization model to select the most relevant instruments for improving the performance of the RP. Burmeister et al. (2010) show that trading constraints are an effective way of regularizing optimization problems so that they produce small RPs. Based on several experiments, they demonstrate that trading costs based on simple statistics provide good performance with minimal computational effort. Burmeister and Mausser (2009) and Burmeister et al. (2010) both present the L_2 and the L_1 methods for the optimization process. However, they do not mention the linear formulation of the L_1 norm that we apply in this paper. Chen and Skoglund (2012a) describe an efficient linear programming approach to cash flow replication. First, they briefly show how cash flow mismatch can be used as an objective function, also mentioning—in a brief footnote—a linear reformulation of the $L1$ norm. For the remainder of their paper they focus on cash flow mismatches as constraints. Their analysis is purely theoretical and they do not provide any case studies that use real-life data. Cambou and Filipovic (2016) provide a novel, dynamic, and path-dependent RP approach for risk-neutral sampling. They focus strongly on the mathematical foundations of their approach and briefly present two numerical examples.

The remainder of this paper is organized as follows: In Section 1.2 we discuss the connection between RPs and economic capital and show the importance of RPs for the life insurance industry. We introduce the mathematical formulation of our RP model in Section 1.3. In Section 1.4 we describe the quality assessment for our model, and discuss practical implications. Section 1.5 provides an example of an actual RP implementation at Zurich and Section 1.6 discusses the results of the RP selection process. Section 1.7 describes how the RP model is integrated into Zurich's framework. Section 1.8 concludes.

1.2 Economic Capital for Life Insurance

In this section, we explain why life insurance companies such as Zurich resort to RP models as a valuation tool for their life insurance portfolios. For this purpose, we first describe how regulatory requirements oblige insurance companies to report economic

capital, which in turn requires them to value their liabilities, such as the life insurance products sold to customers. Second, we argue that (fairly) exact valuations of the cash flows of life insurance products are theoretically feasible but prohibitively expensive in terms of computing time when the number of scenarios is in the order of thousands or more. Insurance companies must, therefore, restrict themselves to approximate valuation tools. We complete this motivation with a brief look at some approximation methods, such as replicating portfolios.

1.2.1 Economic Capital

Financial services firms, such as banks and insurance companies, are expected by customers, regulators, and capital providers to hold a sufficient amount of capital to absorb the financial losses they may incur in times of financial stress. This capital provides security and protection for customers owed money by such firms—for example, to a life insurance holder of a life insurance firm. Porteous and Tapadar (2008) define economic capital as the "amount of capital, or excess assets, that a business requires to ensure that its realistic, or market-value, balance sheet remains solvent, over a specified time horizon, with a prescribed probability or confidence level, following events that are unexpected, yet not so unlikely that they might never occur in practice." In response to market expectations and regulatory regimes, economic capital frameworks have become a common risk management practice in insurance companies.

For insurance companies, the use of economic capital is embedded in the regulatory solvency frameworks of the European Union (Solvency II) and of Switzerland (Swiss Solvency Test). Both require an assessment with a horizon of one year, although they differ in the metric for solvency evaluation—value-at-risk at 99.5 percent for Solvency II and expected shortfall at 99 percent for the Swiss Solvency Test (SST). Within these solvency regimes, insurers must consider various types of risks. Market risk is usually viewed as the largest risk, for both the assets and the liabilities of a life insurance company. As a result, life insurance companies face the challenging task of assessing the effect of market fluctuations on the value of the life insurance products they have sold to customers.

1.2.2 Life Insurance and the Valuation Challenge

Life insurance companies offer a wide range of life insurance products with very different characteristics, (see Daul and Gutiérrez Vidal (2009)). Many products offer policyholders some choices ("embedded options") at later points in time as well as guarantees linked to market-risk factors. Therefore, the future cash flows of such an insurance contract are not known in advance but depend on the future state of certain economic variables, most commonly market variables such as the value of an equity index or the level of interest rates. As a consequence, such life insurance contracts cannot be adequately valued using a simple discounted cash flow approach. Instead, life insurers must value their liabilities in a market-consistent manner, which requires modern asset pricing techniques.

Calculating economic capital for life insurance liabilities that depend on economic variables is a very challenging task, the main problem being the revaluation of the liabilities for a large number of economic scenarios. Economic capital is based on measures of tail risk, like Value at Risk or Expected Shortfall. These metrics cannot be calculated in closed form. They require Monte Carlo simulations of the economic scenarios within the risk horizon (usually one year). For each of these economic scenarios, the company must calculate the value of their liabilities. The calculation of Expected Shortfall, for example, is done by taking the simulated distribution of the company's value (assets minus liabilities) and averaging the value in the $\alpha\%$ of scenarios with most extreme losses (with α being determined by the regulator).

Since insurance companies often use Monte Carlo pricing for the valuation of their liabilities, "nested stochastic modelling," see Feng et al. (2016), are required for economic capital calculations. In a first step, the outer loop, the insurer must define a probability distribution of future economic states of the world, formally referred to as the $\mathbb{P}$ measure; based on this probability distribution, the method chooses scenarios, that is, sample paths representing different economic conditions. The number of such scenarios ranges typically from 10,000 to more than 100,000. In the second step, the inner loop, the method assigns market values to the liabilities based on possible future time paths of cash flows. This sub-task must be executed for each scenario in the outer loop and itself

requires time-consuming calculations.

In arbitrage-free pricing, the price of a financial security can be calculated as the expected value of its cash flows under the risk-neutral measure. For some securities, this expected value leads to a closed-form formula. (For example, the Black–Scholes formula for European-style options.) Unfortunately, life insurance liability models typically do not have closed-form solutions, even for relatively simple insurance products. Instead, calculating the market-consistent value of these products requires solving a complex integral and the most common approach is to use Monte Carlo integration for valuation. Risk-neutral valuations via Monte Carlo integration usually require between 1,000 and 5,000 simulations. But, recall that this calculation must be performed for each economic scenario in the outer loop. As a result, this direct method would require anywhere between 10 million and 500 million simulations (and it would have to be repeated for hundreds of families of insurance products). The size of such a problem is simply too large for a solution to be found in a reasonable amount of time, even with today's supercomputers and computer clusters. While Feng et al. (2016) describe other methods that attempt to reduce running times, there currently exists no reliable version of nested stochastic modelling that would allow life insurance companies to value their liabilities in a matter of hours or a few days.

Although the computing times of nested stochastic modelling methods prohibit their use for economic capital calculations for the entire company, insurance companies do sometimes use them for certain tasks in special types of business, such as reserve requirements for variable annuities.

1.2.3 Replicating Portfolios

Insurance companies—facing short deadlines for reporting financial results to regulators and investors—have been searching for methods to reduce the computational burden of valuing life insurance products. The risk-neutral valuation of complex insurance products that has to be repeated millions of times but for which no closed-form formula or simple numerical approach exists, is the main reason why nested stochastic modelling is pro-

hibitively slow for economic capital calculations. Therefore, a key idea in the valuation of life insurance has been to avoid these nested calculations by using methods that do not require an inner loop. Most (if not all) large insurance companies with many complex lines of business rely, when calculating the arbitrage-free price of their liabilities, on approximation methods that do not require Monte Carlo integration.

Borrowing ideas from the finance literature on pricing financial derivatives, insurers compute portfolios of financial securities that reproduce the payout of their life insurance liabilities as closely as possible. By construction, both the liabilities and these RPs share approximately the same cash flows; therefore, they should have similar values and exhibit similar sensitivities to changes in key economic variables. Insurers like to choose sets of financial securities that—in contrast to life insurance products—can be quickly priced. Such rapid valuation is possible whenever the prices of these securities are given by closed-form solutions or can be quickly computed numerically. The RP is computationally much more efficient than the entire liability portfolio, and can therefore be used in place of the original liability portfolio when performing risk analysis. For example, insurers can obtain market values as often as market data allows, and directly predict the effects of new market conditions on their insurance portfolio.

1.2.4 Alternative Approaches

While Zurich uses RPs to value its life insurance liabilities in its marker risk model, it is worth pointing out that alternative approaches exist. As will become apparent in the detailed description of RPs in the next section, the construction of RPs is essentially a curve-fitting approach. Insurers have chosen different basis functions for this curve fitting, mostly using either polynomials or financial instruments. The first of these approaches is—in the industry—referred to as "curve fitting", while the second is referred to as "replicating portfolios", even though both of them are in essence curve-fitting approaches. One such polynomial-based curve fitting method is based on the Least-Square Monte Carlo approach introduced by Longstaff and Schwartz (2001) for American options. This particular polynomial method is currently the main alternative approach to using RPs.

Zurich prefers to use RPs since they are perceived to be more directly applicable to other tasks beyond the realm of risk management, such as asset-liability management. Having a representation of liabilities in terms of financial instruments makes the task of integrating the risk management framework with investment decision-making much simpler.

There are also alternative ways of implementing RPs. Zurich's RP approach first matches liability cash flows with those of an RP of financial instruments. Subsequently, it uses the valuation formulas of those financial instruments to approximate the main figure of interest—namely, the value of the liabilities. In the life insurance industry, this approach is called "cash flow matching". An alternative approach—called "market value matching" and used by some companies for simpler products—determines the RP in a different manner. This approach first constructs a vector of liability valuations, one for each economic scenario. In a similar way, it constructs vectors of instrument prices. And finally, it matches the liability valuations to a set of instrument price vectors. This more direct approach should not be viewed as an alternative to RPs but rather as a different way of calculating RPs. This method has the advantage of working directly on the function of interest—namely the liability valuations. However, it also has at least two significant disadvantages. First, the calculation of liability valuations is time-consuming. Second, it does not fully utilize all the available information, since the time distribution of cash flows is partially lost. Zurich believes that these disadvantages strongly outweigh the advantage of working directly on the function of interest, and does not, therefore, use this method.

A second alternative, called "present value matching", is an intermediate approach between the two methods described above. It matches cash flows, but not every yearly cash flow individually. It groups cash flows by adding their present value in one or more time buckets and matches those time buckets with the equivalent present values of the financial instruments. While retaining more information than the market value matching approach, this present value matching approach also loses information in the temporal dimension and does not, therefore, fulfill Zurich's requirements.

In sum, regulatory requirements oblige life insurance companies to calculate the value of the life insurance contracts they have sold to customers and to do so under many thousands of scenarios. While a fairly exact valuation (under certain assumptions) of these contracts is possible, the necessary calculations are extremely time-consuming. In fact, they take so much time that insurance companies cannot rely on them in the face of frequent, mandatory reporting deadlines. They must, therefore, use reliable approximate valuation tools. RPs are such a useful tool. The remainder of this paper focuses on the RP model employed by Zurich.

1.3 The RP Model

In this section we present all constituent parts of our RP model. We first discuss the motivation for using cash flow matching and the L_1 norm for the model. Next, we provide detailed information about the relevant samples, scenarios, and simulations used for our model and our available replication instruments. Finally, we present the mathematical formulation of our optimization model and introduce additional constraints.

1.3.1 Cash Flow Matching

Cash flow matching has the objective of constructing a portfolio of financial instruments whose cash flows parallel those of a given liability portfolio as closely as possible. We denote the vector of liability cash flows by LCF and the vector of cash flows of instrument i by ICF_i. The objective, then, is to find the financial instruments' portfolio weights, w_i, that minimize a given norm,

$$||LCF - \sum_i w_i ICF_i||. \tag{1.1}$$

Contrary to life insurance products, whose valuation is very time-consuming, many financial instruments have readily available valuations (prices), V_i. Once we have determined a matching portfolio of financial instruments—that is, the RP with weights w_i for the respective instrument prices V_i—we can approximate the value, L, of the liability portfolio

by the value, $V(RP)$, of the RP,

$$L \approx V(RP) = \sum_i w_i V_i.$$

1.3.2 Norm

The objective function of the cash flow matching approach requires that we specify a particular norm. Throughout this paper, we use the L_1 norm and so minimize the sum of the absolute differences between liability cash flows and the cash flows of the RP.

The simplest approach to the RP optimization—see Daul and Gutiérrez Vidal (2009)—would be to minimize the sum of squared errors, the L_2 norm, without any constraints, because then the optimal portfolio weights are given by the standard Least-Squares analytical solution. However, it is well known that for the detection of outliers (see, for example, Bloomfield and Steiger (1983) and Bektas and Sisman (2010)) the L_1 norm is superior to the L_2 norm. Moreover, Rice and White (1964) show that the L_1 norm is the best choice for estimation if the error distribution has long tails. And so, since the input data sets generated by insurance companies are likely to include extreme cash flows, the L_1 norm appears to be the most appropriate choice for the objective function. In addition, since L_1 minimization problems allow for linear reformulations, the inclusion of linear constraints represents no challenge.

We executed numerical experiments with both norms. The L_2 norm did not perform poorly. However, the model performed better with the L_1 norm in terms of the accuracy of the approximation as measured by the validation test described in section 1.4.3. We believe this is due to the presence of outliers. As we describe in Section 1.3.4, the model uses simulations from a risk-neutral distribution as inputs. Risk-neutral economic scenario generators (ESGs) of the type normally used in the insurance industry have a known tendency to produce many extreme interest rate values within its simulations, in particular when they are based on a log-normal distribution.

While the L_1 norm is more robust with regard to outliers in the risk-neutral distribution, it is not desirable that extreme values in the real-world distribution be completely

ignored, because these values also contribute to determining the capital requirements for the company. To ensure that extreme real-world scenarios are taken into account, we sample not only from the risk-neutral distribution corresponding to current market conditions but also from extreme market conditions, as found in the tail of the real-world distribution.

1.3.3 Time Steps

In the current model, we use yearly cash flows, from years $t = 1, \ldots, T$ (with T normally in the range between 25 and 40 years) as inputs to the optimization problem. Some insurance products still have material cash flows expected after year T, in which case we take the present value of those cash flows and add them as a cash flow in year T.

1.3.4 Samples, Scenarios, and Simulations

So far in our discussion, we have taken the vector of liability cash flows, LCF, as given. This vector, however, must be generated before we can solve the RP model. Obviously, the components of LCF are absolutely essential to the quality of the RP model's results and their usefulness to the insurance company. The LCF components—each representing the value of an input sample—are generated by the ESG and the liability model.

The input samples contain many risk-neutral simulations (in the order of thousands) that belong to several different economic scenarios (in the order of tens). The relationship between the simulations and the economic scenario to which they belong is given by equations of the pricing model. In pricing models used by insurance companies the economic scenario serves as the starting point, itself an parameter to the equations, of the risk-neutral simulations. Let C denote the number of risk-neutral simulations produced by the ESG for each real-world economic scenario. These simulations, when run through the liability model, produce C paths of cashflows LCF for each scenario. The first real-world economic scenario represents the market conditions at the as-of date of the calculations. This is called the "base scenario", $k = 1$. Additionally, other economic scenarios are considered and indexed $k = 2, \ldots, K$. In total we have $S = C \cdot K$ input

samples. Most of the additional scenarios $k = 2, \ldots, K$ are normally defined as a certain sensitivity on the base scenario, that is, a change in only one of the economic variables. For example, one scenario could be a 100-basis points shock to interest rates. Most of the scenarios that are used as input data, are previously calculated for other reporting tasks, specifically the interest rate, equity, property, and volatility sensitivities, all of which are part of Zurich's annual report.

This enables us to explicitly link the cash flows to the market values. The market value of the liability in economic scenario k is computed as

$$L_k = \sum_{c=1}^{C} \sum_{t=1}^{T} p_c LCF_{k,c,t}, \tag{1.2}$$

where p_c is the risk-neutral probability of simulation c, for a large number of simulations C.

1.3.5 Scenario Definition

As the outer loop of the economic capital model is the probability distribution of future economic states of the world, the input samples must reflect these possible future states. To achieve that, we choose a set of scenarios that reflects the support of $\mathbb{P}$. Most—if not all—of the support should be contained within the range covered by the shocks reflected in the scenarios. (Put differently, we regard the RP model's scenarios as being interpolating rather than extrapolating.)

Economic conditions change over time. Insurers therefore also need to update their $\mathbb{P}$ measurement on a regular basis which in turn obliges them to allow for different scenarios. It is convenient to design the scenarios $k = 2, \ldots, K$ as shocks on the base scenario $k = 1$. This issue is relevant for the discussion on quality measurements that we address in Section 1.4, so we illustrate it here with an example.

Example. For simplicity, consider a world with just two interest rates, a short-term and a long-term rate. Further, assume the RP model has no other exogenous economic inputs. Suppose that at the end of a certain year the two interest rates are both at 2%;

we denote the resulting yield curve by $[2\%, 2\%]$. This is the base scenario for the model. Finally, assume that the model includes five additional scenarios, so $K = 6$.

A possible scenario set could be as follows:

$$\{[2\%, 2\%], [3\%, 3\%], [4\%, 4\%], [5\%, 5\%], [1\%, 1\%], [0\%, 0\%]\}.$$

We obtain this set by applying the following parallel movements to the base scenario,

$$\{+1\%, +2\%, +3\%, -1\%, -2\%\}.$$

Such a scenario set could be used for the calibration of an RP. However, the calibration set then may not be used to also assess the quality of the resulting RP since in that case any error measure would yield to an overly optimistic result (see Friedman et al. (2001)). Instead, the quality of the RP has to be assessed with an out-of-sample scenario set. One way to obtain such an out-of-sample set in practice is to wait until the next reporting cycle (usually quarterly or semi-annually). For example, if six months later the interest rates curve is $[2.5\%, 2.5\%]$, the new scenario set—allowing for the same parallel movements as before—would be

$$\{[2.5\%, 2.5\%], [3.5\%, 3.5\%], [4.5\%, 4.5\%], [5.5\%, 5.5\%], [1.5\%, 1.5\%], [0.5\%, 0.5\%]\}.$$

Of course, instead of parallel movements by an absolute value, we may also consider movements of different magnitudes or define shocks in percentage terms. However, it is essential that the scenarios are a good representation of the support of the assumed $\mathbb{P}$ measurement, no matter which approach to scenario generation the insurer is using.

The set of scenarios may change considerably over time. For example, suppose after six months the interest rates are no longer identical but are now $[2.3\%, 2.7\%]$. Using the same movements away from the base scenario as before, the new set of scenarios is

$$\{[2.3\%, 2.7\%], [3.3\%, 3.7\%], [4.3\%, 4.7\%], [5.3\%, 5.7\%], [1.3\%, 1.7\%], [0.3\%, 0.7\%]\}.$$

Unlike the original scenario set, this new set is completely out-of-sample to the calibration process and thus can be the basis of a quality metric. This observation is of importance to our discussion on quality assessments, for which see Section 1.4.

1.3.6 Scenario Structure

Zurich separates the K scenarios into two groups, an in-sample (IS) and an out-of-sample (OS) group. The number of scenarios in these two groups are denoted by K^{IS} and K^{OS}, respectively, with $K = K^{IS} + K^{OS}$. The in-sample group is much larger and contains all the scenarios that reflect changes to the economic parameters ("sensitivities"). The out-of-sample group only contains scenarios that are relevant for evaluating the suitability of the model for economic capital calculations. These are scenarios that are considered to be particularly representative of the market risk distribution.

In each of the K^{IS} in-sample scenarios, the samples (risk-neutral simulations) are divided into two sets: a training set (TR) and a validation set (VA). The training set size is determined by a trade-off between calibration quality and the available computing power. While a larger set provides a higher precision it is also computationally more expensive. The number of samples is constant across the implementation and has to accommodate the most complex liabilities. The training set contains C^{TR} samples for each of the K^{IS} in-sample scenarios. The out-of-sample scenarios are not included in the training set.

1.3.7 Source of Simulations and Out-of-Sample Scenarios

It is important to understand that the creation and evaluation of an RP model is not a stand-alone operation in a large and complex organization such as Zurich. The RP modelling team is not free to choose and generate its input data as it may deem preferable. Instead, it faces significant restrictions with regard to the data it can use for the specification of its inputs to the model. In particular, it may have to use data that has been generated by other organizational units within Zurich for different purposes.

A particular example of the restrictions faced by the RP modelling team is that it must reuse scenario inputs that have already been calculated for other purposes (e.g., embedded

value reporting). Those scenarios are risk-neutral, random, and are not selected for the task of calibrating an RP. The RP team has only a very limited opportunity to request additional runs of the liability models. The provided scenarios are not necessarily the best possible set for the RP calibration, but as we will show in Section 1.4 they are good enough to achieve satisfactory results. In particular, most of the variables are often highly correlated. This feature creates numerical problems and also does not allow for a clean identification of the optimal instruments, as many instruments are highly correlated and thus produce very similar solutions.

While these restrictions are unwelcome in the RP modelling process, they are not as detrimental as one may think. It is important to note that the RP model is ultimately a predictive model since it will be used on scenarios never encountered during the calibration phase. Adequate attention should, therefore, be given to the problem of designing tests that correctly capture the quality of the predictions. Whether the inputs are good enough (in coverage, randomness, etc.) can be judged by the quality of the predictions. Therefore, Zurich's RP team places great emphasis on performing a range of such tests.

One such practice is the division of the available samples into three sets: training, validation, and test (see Bishop (2007)). The training set is used for the calibration of the model's parameters. The validation set is used for selecting the model's hyper-parameters, such as which instrument classes are used. Finally, the test set is reserved for evaluating the chosen model's quality and should not be used to guide model selection.

When applied to the RP model, we immediately notice that the stratified nature of the samples provided means that while we typically have thousands of (risk-neutral) simulations, we only have a few (real world) scenarios. As the second group is the most relevant to the RP model, we have a problem, because separating those K^{OS} scenarios designated as out-of-sample deprives the model of very necessary (and scarce) information. Zurich uses a holdout method for validation, but a more effective approach would be to implement a cross-validation scheme (for more details on these methods see Bishop (2007) or Friedman et al. (2001)). In particular, a so-called leave-p-out cross-validation could provide much better quality estimates for model selection.

1.3.8 Financial Instruments

In its RP model, Zurich uses only financial instruments that are also available in its economic capital model. Since Zurich's economic capital model is provided by an external supplier, its RP modelling group treats the list of available instruments as an exogenous restriction on its RP model. Therefore, we do the same in this paper.

An interesting feature of some of these instruments is that they are investment rules rather than simply financial securities. For example, if the liability cash flows depend on the value of an equity index at a particular point in time, it is insufficient to define the instrument as "holding an equity index at time 0"; rather, we must define this instrument as a function "holding an equity index at time 0 and selling it at time t". Otherwise, the cash flow stream would not be properly defined since "holding an equity index at time 0" does not include any information about the timing of the inflow produced by selling the position and it would not be possible to define equation (1.1). For financial instruments that represent holdings without a natural maturity date, a "hold and sell" function needs to be created for each time step t on the projection horizon or alternatively a future or forward must be used instead of the underlying position. This is necessary due to the "cash flow matching" approach and would not be required under the "market value matching" or "present value matching" approaches described in section 1.2.4.

The following instrument classes have been implemented in Zurich's RP model: zero coupon bonds, equity and property indices, cash indices, interest rate swap payer assets, forward rate agreements, and swaptions. All of these can be "priced" under various market conditions, even though some of them are not traded on deep and liquid financial markets or not available in any market at all. The inclusion of such hypothetical instruments increases flexibility and the quality of the replication results.

We only provide a brief description of these instrument classes here; for more detailed information see Fabozzi (2008). Zero coupon bonds: These bonds do not make periodic coupon (interest) payments. The holder of such a bond generates interest income by buying it below its par value. The face value of the bond is paid at the time of its maturity and the aggregated interest realized for the holder is the par value minus the

purchase price. Equity and property indices: Holding at time zero a certain index that will be sold at a future, predetermined point in time. The cash flow, which only occurs at this predefined point in time, is based on the value of the underlying index. The equity index is usually the most important equity index in the currency of the liabilities (e.g. for liabilities in CHF the equity index is usually the SMI). The property index can be regarded as a synthetic real estate index in the currency of the liabilities. Cash indices are cash accounts that accumulate annual interest until maturity and then pay out. Interest rate swap payer assets pay a fixed interest payment to another party at designated dates over the life of a specific contract. They are an efficient hedge against short-term interest rate increases (see Fabozzi and Buetow (2008a)). Forward rate agreements pay a fixed interest payment to another party at one designated date (as opposed to many dates in the case of the swap). Swaptions are options on interest rate swaps. A payer (receiver) swaption grants its holder the right to enter into an interest rate swap in which he or she pays (receives) a fixed rate and receives (pays) a floating rate (see Fabozzi and Buetow (2008b)).

1.3.9 Optimization Model—Linear Reformulation and Constraints

Following our qualitative description of Zurich's RP model, we now provide a mathematical formulation of the underlying optimization model. We first state the objective function and provide a linear reformulation. Subsequently, we introduce additional constraints to the linear programming model.

1.3.10 Model Specification

We first introduce some necessary notation for our model. We try to replicate the discounted cash flows of life insurance liabilities using a set of different instrument classes, $a = 1, \ldots, A$. Each of these instrument classes contains a set of I_a, $a = 1, \ldots, A$ candidate instruments. The discounted cash flows occur at discrete times $t = 1, \ldots, T$, where $t = 1$ is the first year and T is the final year of the planning horizon. For the RP optimization, we use the in-sample scenarios $k^{IS} = 1, \ldots, K^{IS}$. Within each of these scenarios we use

the training simulations $c^{TR} = 1, \ldots, C^{TR}$. Table 1.1 summarizes the indices for our model.

Table 1.1: *List of Indices*

Notation	Name	Notation	Name
$k = 1, \ldots, K$	Scenarios	IS	In-sample
$c = 1, \ldots, C$	Simulations	OS	Out-of-sample
$t = 1, \ldots, T$	Time points	TR	Training
$a = 1, \ldots, A$	Instrument classes	VA	Validation
$i = 1, \ldots, I_a$	Instruments		

This table presents the notation for the indices in our optimization model.

In Section 1.3.4 we described the calculation of the discounted liability cash flows $LCF_{k^{IS},c^{TR},t}$ in scenario k^{IS} for simulations c^{TR} at time t, and the (discounted) instrument cash flows $ICF_{k^{IS},c^{TR},t,a,i}$ for instrument i in instrument class a in scenario k^{IS} for simulation c^{TR} at time t. Furthermore, the values of the liabilities $\overline{L}_{k^{IS}}$ and the RP $\overline{RP}_{k^{IS}}$ are, respectively, approximated for each scenario k^{IS} with the following equations:

$$\overline{L}_{k^{IS}} = \frac{1}{C^{TR}} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} LCF_{k^{IS},c^{TR},t} \tag{1.3}$$

$$\overline{RP}_{k^{IS}} = \frac{1}{C^{TR}} \sum_{a=1}^{A} \sum_{i=1}^{I_a} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} ICF_{k^{IS},c^{TR},t,a,i} w_{a,i}, \tag{1.4}$$

These cash flows are now the parameters for an optimization model.

We minimize the sum of the absolute differences between the discounted liability cash flows and the RP cash flows for all years t and all training simulations c^{TR} in all in-sample scenarios k^{IS},

$$min_w \sum_{k^{IS}=1}^{K^{IS}} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} \left| LCF_{k^{IS},c^{TR},t} - \sum_{a=1}^{A} \sum_{i=1}^{I_a} w_{a,i} ICF_{k^{IS},c^{TR},t,a,i} \right|, \tag{1.5}$$

where the decision variables $w_{a,i}$ are the holdings in each of the $i = 1, 2, \ldots, I_a$ instruments in each instrument class $a = 1, 2, \ldots, A$.

We replace each decision variable $w_{a,i}$ by the difference of a nonnegative long position $l_{a,i}$ and a nonnegative short position $s_{a,i}$,

$$w_{a,i} = l_{a,i} - s_{a,i}, \quad l_{a,i}, s_{a,i} \geq 0, \quad \forall\ a, i. \tag{1.6}$$

This variable change proves very helpful in the formulation of the additional constraints that we present in Section 1.3.12.

1.3.11 Linear Reformulation

To obtain numerically tractable linear programs we employ a linear reformulation of our optimization problem. This linear reformulation has not yet been applied to RPs but appears to be widely used in portfolio optimization. For example, Rudolf et al. (1997) apply it to tracking error minimization. According to Feinstein and Thapa (1993), this approach is highly efficient due to its relatively small number of additional auxiliary constraints. For this reformulation, we introduce for each term in the original L_1 objective function the positive deviation $y^+_{k^{IS},c^{TR},t}$ and the negative deviation $y^-_{k^{IS},c^{TR},t}$ as auxiliary variables and minimize their sum in the new linear objective function,

$$min_{l,s,y^+,y^-} \sum_{k^{IS}=1}^{K^{IS}} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} \left(y^+_{k^{IS},c^{TR},t} + y^-_{k^{IS},cTR,t} \right). \tag{1.7}$$

Then we relate the auxiliary variables to the terms in the original objective function via constraints,

$$y^+_{k^{IS},c^{TR},t} - y^-_{k^{IS},c^{TR},t} = LCF_{k^{IS},c^{TR},t} - \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{k^{IS},c^{TR},t,a,i}(l_{a,i} - s_{a,i}) \quad \forall\, k^{IS}, c^{TR}, t \tag{1.8}$$

$$y^+_{k^{IS},c^{TR},t}, y^-_{k^{IS},c^{TR},t} \geq 0 \quad \forall\ k^{IS}, c^{TR}, t \tag{1.9}$$

Konno and Yamazaki (1991) present a different option for a linear reformulation, which we present in Appendix 1.9. This formulation leads to significantly longer running times than the one used for the example.

$$l_{a,i}, s_{a,i} \geq 0 \quad \forall\, a, i. \tag{1.10}$$

1.3.12 Additional Constraints

Seemann (2009) argues that an RP may need to have additional "nice" properties for it to be helpful in the context of a thorough analysis of a life insurance portfolio. However, these properties are not reflected in the objective function—that is, the simple minimization of the L_1 (or any other) norm may not lead to an RP that has such favorable properties. For the computation of a truly helpful RP, we therefore introduce additional constraints to the model. In Zurich's experience, two sets of constraints are very helpful. Burmeister and Mausser (2009) explain that an RP should have a relatively small number of instruments because a small RP is relatively quick to price, easy to interpret in relation to the liabilities, and suited to replicating the liabilities across a wide range of market conditions. In order to limit the number of instruments in the optimal solution, we introduce a set of constraints to directly control the size of the complete RP. These constraints have the additional benefit of limiting the absolute magnitude of the individual portfolio positions. From a risk perspective, Chen and Skoglund (2012a) advise not only that the cash flow (mis)match be considered in general, but also that special attention be paid to potential outliers for extreme scenarios. For this purpose, we also impose a set of constraints to ensure a good cash flow match for a broad range of future events.

The first set of constraints is established to limit the total number of instruments in the portfolio and to avoid so-called offsetting problems. Offsetting refers to the consequences of the well-known multi-collinearity problem from regression analysis in the context of RPs. Such an RP exhibits very large long positions in some instruments, which are offset by very large short positions in instruments with almost collinear payoffs across the scenarios. While such an RP may offer a slightly improved in-sample fit compared to an RP restricted by bounds, it usually leads to significantly worse out-of-sample fits and

it is, therefore, undesirable.

We impose the following linear constraint for each instrument class a:

$$\sum_{i=1}^{I_a}(l_{a,i}+s_{a,i}) \leq \frac{nl}{nv_a c^{TR}} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} LCF_{1,c^{TR},t} \qquad \forall a, \tag{1.11}$$

with the parameters nl and nv_a.

The notional limit, nl, determines the largest permitted value for the notional values of an instrument class relative to the discounted liability cash flows in the base case. This set of constraints provides bounds on the sums of the absolute values of the short positions, $s_{a,i}$, and long positions, $l_{a,i}$, of every instrument class. A strength of this approach is that it sets a hard bound on the sum of long and short positions of each instrument class. Other methods for reducing offsetting—such as the alternative form of regularization presented in Appendix 1.10—only provide soft bounds and need additional calibration effort for setting the penalty parameter $\lambda_{a,i}$ appropriately.

The so-called notional value nv_a for the instrument class a is the par value of a financial instrument in this class. (For example, the par value of a bond is defined as the amount that the issuer agrees to repay at the maturity date to the bondholder.) These notional values can be different depending on the instrument class, but they are the same for all the instruments in each class. This is purely a matter of market convention and convenient to know the size of an instrument unit.

If an RP is to be useful, it is essential to ensure an adequate replication of the liabilities for every economic shock that affects the liabilities. In our application, for example, the instruments have to provide an excellent match for the liabilities for all future changes in the interest rate, equity index, and property index that we consider in our scenarios. We try to ensure this match by relating the non-base scenarios to the base scenarios. We want the discounted instrument cash flows in a non-base scenario to differ from the discounted instrument cash flows in the base scenario set in the same way as the discounted liability cash flows in a non-base scenario set differ from the discounted liability cash flows in the

base scenario. For example, if the liabilities increase by 20 percent compared to the base scenario due to an interest rate decrease, we want the RP to also increase by 20 percent compared to the base case. To limit these scenario movements between the base scenario $k^{IS} = 1$ and the in-sample non-base scenarios $k^{IS} = 2, \ldots, K^{IS}$ we first introduce a new set of variables. For each of the non-base scenarios the variable $u_{k^{IS}}$ is defined as:

$$u_{k^{IS}} = \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{k^{IS},c^{TR},t,a,i}(l_{a,i} - s_{a,i}) - \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{1,c^{TR},t,a,i}(l_{a,i} - s_{a,i}) - \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} LCF_{k^{IS},c^{TR},t} + \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} LCF_{1,c^{TR},t}. \tag{1.12}$$

Then we introduce an upper bound *adub* and a lower bound *adlb* for $u_{k^{IS}}$:

$$adlb * \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} LCF_{1,c^{TR},t} \leq u_{k^{IS}} \leq adub * \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} LCF_{1,c^{TR},t}. \tag{1.13}$$

The upper and lower bound (1.13) must hold for $k^{IS} = 2, \ldots, K^{IS}$.

This completes the development of our linear optimization model. The objective function in (1.7) is minimized subject to the constraints (1.8)–(1.13). In a final step of our RP optimization process, we calculate the trimmed solution for every optimal solution.

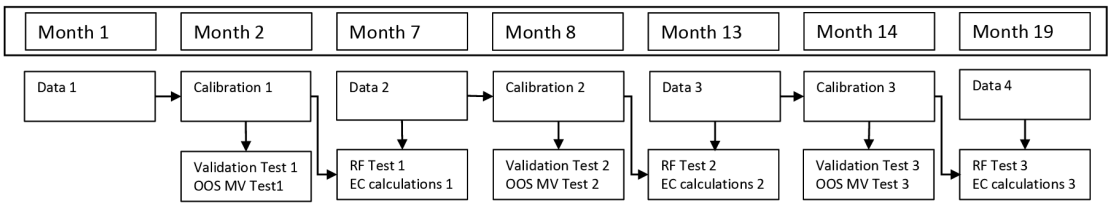
1.4 Quality Assessment

As RPs are a proxy model for capital calculations, their most important characteristic is their ability to reproduce liability valuations with a low level of error. For this model to be successful for Zurich, it should provide a good approximation to economic capital as it would have been calculated by liability models.

The first step of the quality assessment is a relative evaluation, carried out using what we call the validation test and out-of-sample test. Different possible RPs are measured against each other in order to identify which of them provides the better fit to the data.

The detailed trimming description is presented in Appendix 1.11.

Figure 1.1: *Time line for the Quality Assessment*



This figure provides an overview of the timeline for the RP quality assessment at Zurich.

An iterative process is put in place to evaluate different hyper-parameters and measure the quality of the RPs these parameters produce. This step ends in the selection of the optimal RP but does not by itself tell us whether the RP model is a successful proxy. It is only later, once more information is available, that an absolute evaluation is performed. We call this test the "roll forward test". In this step we try to find out whether this RP is good enough for capital calculations. The "roll forward test" is the measure of the model's success.

1.4.1 Time line for the Quality Assessment

Figure 1.1 shows the time line for the RP quality assessment at Zurich in generic months. Every six months (Month 1, Month 7, Month 13, etc.) new data is provided. In the next month (Month 2, Month 9, Month 16, etc.) the model is calibrated based on this data. During the calibration, Zurich uses a validation test to select an optimal RP. After the validation test, the company assesses and confirms the quality of the optimal RP with our out-of-sample market value movement test. Five months later (in Month 7, Month 13, Month 19, etc.), new data is used to assess the quality of the calibrated portfolio with a "roll forward" test. Afterwards, economic capital calculations are conducted for the portfolio that was calibrated six month ago.

1.4.2 First Stage of the Quality Assessment

With our LP model we are able to quickly calculate different RPs depending on the chosen constraint parameters. The next crucial step is to assess the relative quality of

these RPs.

In the first stage of the quality assessment—which takes places immediately after the calibration—we use a validation test and an out-of-sample market value movement test to select and confirm one optimal RP.

1.4.3 Validation Test

During the calibration process we use different values for *nl*, *adub*, and *adlb* for the additional constraints we presented in Section 1.3.12 to calculate several RPs. Warm starts—based on previous computations—ares used to improve the performance in the calibration process. Finally, we select the one out of these RPs (which we refer to as the "optimal RP") that performs best when market variables such as interest rates, equity indices, and property indices develop in unexpected ways.

We measure best performance with a validation test. This test ensures that we are not over-fitting our weights for certain simulations in a scenario. We assess the performance of our RP in each scenario with validation simulations that are different from the training simulations used for the optimization. Ideally, our non-base, in-sample scenarios differ as much from the base scenario for these validation simulations as they do in the training data set, since we want to ensure that we have captured the changes in market variables appropriately in the optimization. The validation movement $VM_{k^{IS}}$ for all non-base, in-sample scenarios $k^{IS} = 2, \ldots, K^{IS}$ is denoted by

$$VM_{k^{IS}} = \left(\sum_{c^{VA}=1}^{C^{VA}} \sum_{t=1}^{T} \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{k^{IS},c^{VA},t,a,i}(w_{a,i}) - \sum_{c^{VA}=1}^{C^{VA}} \sum_{t=1}^{T} \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{1,c^{VA},t,a,i}(w_{a,i}) - \sum_{c^{VA}=1}^{C^{VA}} \sum_{t=1}^{T} LCF_{k^{IS},c^{VA},t} + \sum_{c^{VA}=1}^{C^{VA}} \sum_{t=1}^{T} LCF_{1,c^{VA},t} \right) / \sum_{c^{VA}=1}^{C^{VA}} \sum_{t=1}^{T} LCF_{1,c^{VA},t}. \tag{1.14}$$

The right-hand side of (1.14) closely resembles the right-hand side of the constraints (1.13) in our LP. However, we substitute the discounted liability cash flows $LCF_{k^{IS},c^{TR},t}$ of the training data set by the discounted liability cash flows $LCF_{k^{IS},c^{VA},t}$ of the validation data

set, and the discounted training instrument cash flows $ICF_{k^{IS},c^{TR},t,a,i}$ by the discounted validation instrument cash flows $ICF_{k^{IS},c^{VA},t,a,i}$. As a result, the values $VM_{k^{IS}}$ may violate the bounds *adub* and *adlb* and so may give us an indication of how poorly the optimal RP behaves for different simulations.

1.4.4 Out-of-sample Market Value Movement Test

After the validation test we confirm the optimal RP with a market value test. This test includes all out-of-sample scenarios, $k^{OS} = 1, \ldots, K^{OS}$. These scenarios consider far more extreme events than the in-sample scenarios. Furthermore—in contrast to the validation test presented in the previous section—these scenarios are not included in the training data set and we use all generated simulations for the validation.

We compare the market value movements of all out-of-sample scenarios with the market value movement of the base scenario $k = 1$. The calculations are based on all the simulations that were generated for the base scenario (and not just a portion of them). Both, the instruments and the liabilities are priced with this approach. While it would be possible to price the instruments with a closed-form formula, we price it in the same way as the liabilities (that cannot be priced with a closed-form formula) to ensure methodological consistency between liabilities and the RP. A consistent method for both sides is necessary for reliable error measures.

$$MV_{k^{OS}} = \frac{\sum_{a=1}^{A}\sum_{i=1}^{I_a}(MVI_{k^{OS},a,i}(w_{a,i}) - MVI_{base,a,i}(w_{a,i})) - (MVL_{k^{OS}} - MVL_{base})}{MVL_{base}}, \tag{1.15}$$

where $MVI_{base,a,i}$ and $MVI_{k^{OS},a,i}$ are the market values for instrument class a and instrument i in the base scenario and for all out-of-sample scenarios $k^{OS} = 1, \ldots, K^{OS}$, respectively. MVL_{base} and $MVL_{k^{OS}}$ are the values for the market value liabilities in the base scenario and for all out-of-sample scenarios $k^{OS} = 1, \ldots, K^{OS}$ based on all simulations generated for the respective scenario.

1.4.5 Second Stage of the Quality Assessment

The second stage of the quality assessment takes places approximately six months after the calibration. At this point in time, a new set of scenarios $k^{NE} = 1, \ldots, K^{NE}$ has been received as part of the regular process of capital calculations. These scenarios are produced to serve as the calibration input for the next set of replicating portfolios, but they are also used to measure the quality of the existing replicating portfolios.

In the test during this stage we assess whether the RP provides a good approximation of the economic capital concerned. For this purpose, we estimate the error in the RP estimation of the economic capital. This RP-estimated economic capital, $\widehat{EC}$, is the mean of the estimated surplus (assets minus replicating portfolios) minus the expected shortfall of the estimated surplus.

$$\widehat{EC} = E[A - RP] - ES[A - RP],$$

while the true economic capital is the mean of the true surplus (assets minus liabilities) minus the expected shortfall of the true surplus

$$EC = E[A - L] - ES[A - L].$$

The order of the terms above, total mean minus tail mean rather than tail mean minus total mean, is given by the sign convention of EC being positive.

The error is then

$$\widehat{EC} - EC = (E[A - RP] - ES[A - RP]) - (E[A - L] - ES[A - L]), \tag{1.16}$$

which represents by how much we are underestimating or overestimating the expected shortfall of gains and losses on the asset/liability portfolio. For a given scenario X, $L(X)$ and $RP(X)$ are the value of the liabilities and the RP, respectively. $A(X)$ is the value of the assets for scenario X. When sufficiently clear from the context, L and RP are used for $L(X)$ and $RP(X)$ to simplify the notation.

Since in this section we work with market values at time zero, X does not need a subscript for time, but does need one for the different economic variables that are used as inputs for the valuations, $X_1, \ldots, X_p, \ldots, X_P$, as well as a superscript for distinguishing between different scenarios $X^{(1)}, \ldots, X^{(k)}, \ldots, X^{(K^{NE})}$. This notation is used in all subsequent formulas.

An exact error calculation in Equation (1.16) would require a calculation of $ES[A - L]$, which we cannot do for the reasons explained in Section 1.1 and Section 1.2. Therefore, we can only provide an approximation that is limited to the available points of $L(X)$. In our first step we find the conditions that enable us to eliminate the dependency on knowing $L(X)$ for all X, in particular its exact behavior under the tail of $A(X) - L(X)$, which is denoted as $\mathcal{T}_{A-L}$:

$$\mathcal{T}_{A-L} = \{X \mid A(X) - L(X) < \alpha_{A-L}\},$$

where α_{A-L} is defined by $P\{A(X) - L(X) < \alpha_{A-L}\} = \alpha$.

Rearranging (1.16) we obtain

$$\begin{aligned}
\widehat{EC} - EC &= (E[A - RP] - ES[A - RP]) \\
&\quad - (E[A - L] - ES[A - L]) \\
&= (E[A - RP] - E[A - RP \mid X \in \mathcal{T}_{A-RP}]) \\
&\quad - (E[A - L] - E[A - L \mid X \in \mathcal{T}_{A-L}]) \\
&= (E[A - RP] - E[A - L]) \\
&\quad - (E[A - RP \mid X \in \mathcal{T}_{A-RP}] - E[A - L \mid X \in \mathcal{T}_{A-L}])),
\end{aligned} \tag{1.17}$$

which is the error of the mean minus the error of the expected shortfall.

The error of the mean can be re-expressed to eliminate the dependency on $A(X)$,

$$(E[A - RP] - E[A - L])) = E[L - RP]. \tag{1.18}$$

If $RP(X)$ is order-preserving on $L(X)$—meaning that $\mathcal{T}_{A-RP} = \mathcal{T}_{A-L}$ holds— then the error of the expected shortfall can be similarly re-expressed to eliminate $A(X)$ as

$$\begin{aligned} & E[A - RP \mid X \in \mathcal{T}_{A-RP}] - E[A - L \mid X \in \mathcal{T}_{A-L}] \\ = & E[A - RP \mid X \in \mathcal{T}_{A-RP}] - E[A - L \mid X \in \mathcal{T}_{A-RP}] \\ = & E[L - RP \mid X \in \mathcal{T}_{A-RP}]. \end{aligned} \tag{1.19}$$

We combine (1.18) and (1.19) with (1.16) and (1.17) and obtain

$$\widehat{EC} - EC = E[RP - L \mid X \in \mathcal{T}_{A-RP}] - E[RP - L], \tag{1.20}$$

where the expectations are taken over the approximation error $RP(X) - L(X)$, which we call $\mathcal{E}(X)$. If the RP were an entirely incorrect proxy, the tail would not be preserved. To mitigate this possibility, Zurich applies other tests in the selection of the RP as described in Section 1.4.2.

To calculate $E[\mathcal{E}(X) \mid X \in \mathcal{T}_{A-RP}]$ and $E[\mathcal{E}(X)]$, we approximate $\mathcal{E}(X)$ using its Taylor expansion

$$\mathcal{E}(X) = \mathcal{E}(X^{(a)}) + (X - X^{(a)})\nabla\mathcal{E}(X^{(a)}) + \frac{1}{2}([X - X^{(a)}]^T)\nabla^2\mathcal{E}(X^{(a)})[X - X^{(a)}] + \dots.$$

If we assume the error is multivariate linear in X, then

$$\mathcal{E}(X) = \mathcal{E}(X^{(a)}) + (X - X^{(a)})\nabla\mathcal{E}(X^{(a)}). \tag{1.21}$$

The choice of $X^{(a)}$ and the calculation of $\nabla\mathcal{E}(X^{(a)})$ lead to several choices and practical challenges, which we discuss in the next section.

1.4.6 Quality Measurement in Practice

The most straightforward way of combining Equations (1.20) and (1.21) is to take a global linearity assumption and not make any allowance for the possible error in case the

error function is not linear. This leads to

$$E[\mathcal{E}(X) \mid X \in T_{A-RP}] = \mathcal{E}(X^{(a)}) + (\widetilde{X} - X^{(a)})\nabla\mathcal{E}(X^{(a)}) \tag{1.22}$$

where $\widetilde{X} = [E[X_1 \mid X \in \mathcal{T}_{A-RP}], \ldots, E[X_p \mid X \in \mathcal{T}_{A-RP}]]$, and

$$E[\mathcal{E}(X)] = \mathcal{E}(X^{(a)}) + (\bar{X} - X^{(a)})\nabla\mathcal{E}(X^{(a)}) \tag{1.23}$$

where $\bar{X} = [E[X_1], \ldots, E[X_p]]$.

Combining (1.20) with (1.22) and (1.23) we obtain

$$\widehat{EC} - EC = \mathcal{E}(X^{(a)}) + (\widetilde{X} - X^{(a)})\nabla\mathcal{E}(X^{(a)}) - (\mathcal{E}(X^{(a)}) + (\bar{X} - X^{(a)})\nabla\mathcal{E}(X^{(a)})) = (\widetilde{X} - \bar{X})\nabla\mathcal{E}(X^{(a)}).$$

In this fully linear case, the calculation of $\nabla\mathcal{E}$ can be performed at any point $X^{(a)}$ and using any interval for the finite difference.

A variation of the fully linear approach involves a separate linearization of (1.22) and (1.23), around $\widetilde{X}$ and $\bar{X}$, respectively. This would improve the robustness against non-linearity in $\mathcal{E}(X)$, since it only requires a locally linear assumption:

$$\begin{aligned} \widehat{EC} - EC &= E[\mathcal{E}(\widetilde{X}) + (X - \widetilde{X})\nabla\mathcal{E}(\widetilde{X}) \mid X \in \mathcal{T}_{A-RP}] - E[(\mathcal{E}(\bar{X}) + (X - \bar{X})\nabla\mathcal{E}(\bar{X}))] \\ &= \mathcal{E}(\bar{X}) - \mathcal{E}(\widetilde{X}). \end{aligned} \tag{1.24}$$

This option leads to an appealing and intuitive formula: the error in the economic capital is the error in the full distribution minus the error in the tail. This formula requires to be able to define $\widetilde{X}$ before requesting the information from the liability models. In practice, however, a decision on the required inputs has to be made before $\widetilde{X}$ is known. One possible solution is to use $\widetilde{X}$ from a previous run. A second possibility is to use Equation (1.24) and the linear assumption to estimate $\mathcal{E}(\widetilde{X})$ and $\mathcal{E}(\bar{X})$ from the data available in S^{NE}, the set of K^{NE} scenarios generated as part of the next reporting period (usually $K^{NE} = K$ unless the company decides to expand the input set). These scenarios

are entirely out-of-sample and are generated six months after the scenarios that are used for the model calibration and first-stage testing. In case we have sensitivities to all X_p within our K^{NE} scenarios—these are scenarios with only one X_p variable changed at a time, and a magnitude of that sensitivity sufficiently large to place it within $\mathcal{T}_{A-RP}$—we can use this set of sensitivities, $\{(X^{(k)}, \mathcal{E}(X^{(k)}))\}_{k=1}^{K^{NE}}$, in Equation (1.21) to calculate $(X - X^{(a)})\nabla\mathcal{E}(X^{(a)})$. This second option within the locally linear approach is the one that we use and we explain further in the following paragraphs.

The starting point are equations (1.22) and (1.24), which we combine to obtain

$$\widehat{EC} - EC = \mathcal{E}(\bar{X}) - \mathcal{E}(\widetilde{X}) = \mathcal{E}(X^{(1)}) - (\mathcal{E}(X^{(a)}) + (\widetilde{X} - X^{(a)})\nabla\mathcal{E}(X^{(a)})).$$

We make use of the fact that S^{NE} includes the mean, base scenario for $k^{NE} = 1$ and therefore $\mathcal{E}(\bar{X}) = \mathcal{E}(X^{(1)})$.

As a first step we have to find the reference point $X^{(a)}$ of the linearization using the new scenarios S^{NE}. For each dimension X_p in X we identify a scenario $X^{(k^{NE})}$ within the available scenarios, where the shock to that dimension of X in that scenario, $X_p^{(k^{NE})}$, is large enough to make it a member of T_{A-RP} (if we only considered that dimension). Once we have identified those scenarios, we can form a set of dimension-scenario pairs $\{X_p, \mathcal{E}(X^{(\mathcal{K}(p))})\}$, with its implicit mapping $\mathcal{K}$: dimension p $\rightarrow$ scenario k that allows us to build the vector $X^{(a)}$,

$$X^{(a)} = [X_1^{\mathcal{K}(1)}, \ldots, X_P^{\mathcal{K}(P)}].$$

In the second step, we build the gradient $\nabla\mathcal{E}(X^{(a)})$. This is calculated as the finite difference between $X_p^{\mathcal{K}(p)}$ (an scenario with a shock larger than $\widetilde{X}_p$) and the closest point along that dimension which we will call $X_p^{\mathcal{N}(p)}$ (an scenario with a shock smaller than $\widetilde{X}_p$). In this way, $\widetilde{X}$ (the average tail shock) lies between the scenarios provided by $\mathcal{K}$ and $\mathcal{N}$. With these scenarios we calculate the p-th element of $\nabla\mathcal{E}$ as

$$\nabla\mathcal{E}_p(X^{(a)}) = \frac{\mathcal{E}(X_p^{\mathcal{K}(p)}) - \mathcal{E}(X_p^{\mathcal{N}(p)})}{X_p^{\mathcal{K}(p)} - X_p^{\mathcal{N}(p)}}.$$

In the final step, we write explicitly the sum across dimensions instead of the vector

notation and obtain

$$\begin{aligned}\widehat{EC} - EC &= \mathcal{E}(\bar{X}) - \sum_p \mathcal{E}(\widetilde{X}_p) \\ &= \mathcal{E}(X^{(1)}) - \sum_p \mathcal{E}(X_p^{(a)}) + (\widetilde{X}_p - X_p^{(a)})\Delta\mathcal{E}(X_p^{(a)}) \\ &= \mathcal{E}(X^{(1)}) - \sum_p \mathcal{E}(X_p^{(a)}) + (\widetilde{X}_p - X_p^{(a)})\frac{\mathcal{E}(x_p^u) - \mathcal{E}(x_p^l)}{x_p^u - x_p^l}\end{aligned} \tag{1.25}$$

where $x_p^u = X_p^{\mathcal{K}(p)}$ and $x_p^l = X_p^{\mathcal{N}(p)}$.

Based on Equation (1.25), and after re-arranging the terms, we formulate our roll forward test. In this test the total error is split into the contribution from each risk factor (dimension), each given by $R_p(\widetilde{X})$:

$$\widehat{EC} - EC = \mathcal{E}(\bar{X}) - \mathcal{E}(\widetilde{X}) = \sum_p R_p(\widetilde{X}),$$

with

$$\begin{aligned}R_p(\widetilde{X}) = \Bigg[&\left(\frac{x_u - \widetilde{X}_p}{x_u - x_l}\right)\left((RL_{x_l} - RL_1) - \sum_{a=1}^{A}\sum_{i=1}^{I_a}(RI_{x_l,a,i}(w_{a,i}) - RI_{1,a,i}(w_{a,i}))\right) \\ &+ \left(\frac{\widetilde{X}_p - x_l}{x_u - x_l}\right)\left((RL_{x_u} - RL_1) - \sum_{a=1}^{A}\sum_{i=1}^{I_a}(RI_{x_u,a,i}(w_{a,i}) - RI_{1,a,i}(w_{a,i}))\right)\Bigg] / RL_1),\end{aligned} \tag{1.26}$$

where RL_1, RL_{x_l}, and RL_{x_u} are the values for the liabilities in the base scenario and for the scenarios with a smaller and larger magnitude than x in the set of new scenarios $k^{NE} = 1, \ldots, K^{NE}$, respectively, and $RI_{1,a,i}$, $RI_{x_l,a,i}$, and $RI_{x_u,a,i}$ are the roll forward values for instrument class a and instrument i in the base scenario and for the scenarios with a smaller and larger magnitude than x, respectively.

Note that since $EC(X) = EC(X + k)$ for any constant k, the quality metric described is not affected by constant errors $\mathcal{E}(X)) = k$.

The quality measurement of the RPs is essential to an evaluation of how successfully the model fulfills its intended purpose. Zurich has been using the procedure described in

this section for four years now with satisfactory results. At the group level, the error has always been below the target error level. These results show that—under the assumptions described above—the model is well-suited for approximating the liabilities for economic capital calculations.

1.5 RP Implementation Example

In this section we provide an example of an actual implementation of the described RP model at Zurich. For this example, we will apply the model to one specific product, a so-called Guaranteed Annuity Option (GAO). This product is a subset of Zurich's USD 209 billion life insurance liabilities (measured by fair value reserves as of December 2016, Zurich Insurance Group (2017)). All values are normalized to protect confidential information.

First, we specify the scenarios, simulations, and time steps used for our example. Second, we discuss the structure of the liabilities, which we have to replicate with our candidate instruments. Third, we describe the properties of these candidate instruments.

1.5.1 Specification of Scenarios, Simulations, and Time Steps

Table 1.2 provides an overview of the total simulations and the simulations for each subset per scenario for our specific example. In total we have 15 scenarios; of these, 13 are used in-sample while the remaining two scenarios are only used for out-of-sample tests. For each in-sample scenario we have 200 simulations. We use 100 of these simulations for the optimization and 100 for the validation test. For the two out-of-sample scenarios we have

Table 1.2: *Scenarios and Simulations*

Scenario	Total	Training	Validation
k	C	C^{TR}	C^{VA}
$k^{IS} = 1, 2, \ldots, 13$	200	100	100
$k^{OS} = 1, 2$	1000	0	1000

This table presents the number of simulations for each scenario in total and for each subset.

1,000 simulations and use all of them for the validation test.

Table 1.3 describes the economic conditions characterizing the 15 scenarios. In Section 1.3.4 we explained that the set of scenarios has to represent a plausible range of economic conditions and contain sufficient shocks to all the economic variables. For the data in our specific example, the development of the interest rate and the equity performance are the essential real-world economic factors.

Table 1.3: *List of Scenarios*

Notation	Feature	Notation	Feature
$k^{IS} = 1$	Base	$k^{IS} = 9$	EI large down
$k^{IS} = 2$	IR small down (−1%)	$k^{IS} = 10$	EI vol up (+25%)
$k^{IS} = 3$	IR small up (+1%)	$k^{IS} = 11$	EI vol down (−25%)
$k^{IS} = 4$	IR large down	$k^{IS} = 12$	PI small down (−10%)
$k^{IS} = 5$	IR large up	$k^{IS} = 13$	PI large down
$k^{IS} = 6$	IR vol down (−25%)	$k^{OS} = 1$	2008 shock
$k^{IS} = 7$	IR vol up (+25%)	$k^{OS} = 2$	Tail shock
$k^{IS} = 8$	EI small down (−10%)		

This table presents the scenarios for the optimization model and the tests. IR = interest rate, EI = equity index, PI = property index, vol = volatility.

The development of equity markets determines the amount Zurich owes to a policy holder once a given policy matures. The long-term interest rates and the interest rate volatility are relevant for the annuity payments Zurich has to make from this point on. Lower interest rates and higher interest rate volatility must lead to smaller annuity payments. This could be problematic for the GAOs that Zurich offers to the policy holder. Under such a GAO, Zurich guarantees that it will convert an insurance holder's accumulated funds to a life annuity at a fixed rate once the policy matures (see Boyle and Hardy (2003) and Pelsser (2003)).

Our in-sample scenarios consider up and down movements and changes in the volatility of the interest rate and the equity index. We reuse in-sample Scenarios 2, 3, 8, and 9 from Zurich's embedded value reporting. In-sample Scenarios 4–7 and 10–11 simulate a range

of shocks to cover a wide (but not extreme) range of economic conditions. Typically, Scenarios 12–13 cover changes in the property index. However, for this specific example, changes in the property index are not relevant and Scenarios 12 and 13 are set equal to the base scenario but are based on different simulations than the 100 simulations used for the base scenario. The two out-of-sample scenarios simulate very extreme conditions. The first simulates the shock of the financial crisis in 2008. It is common to use this shock—with small differences in the exact specification—among life insurance companies. The second out-of-sample scenario is taken from the market risk distribution and is specific to Zurich.

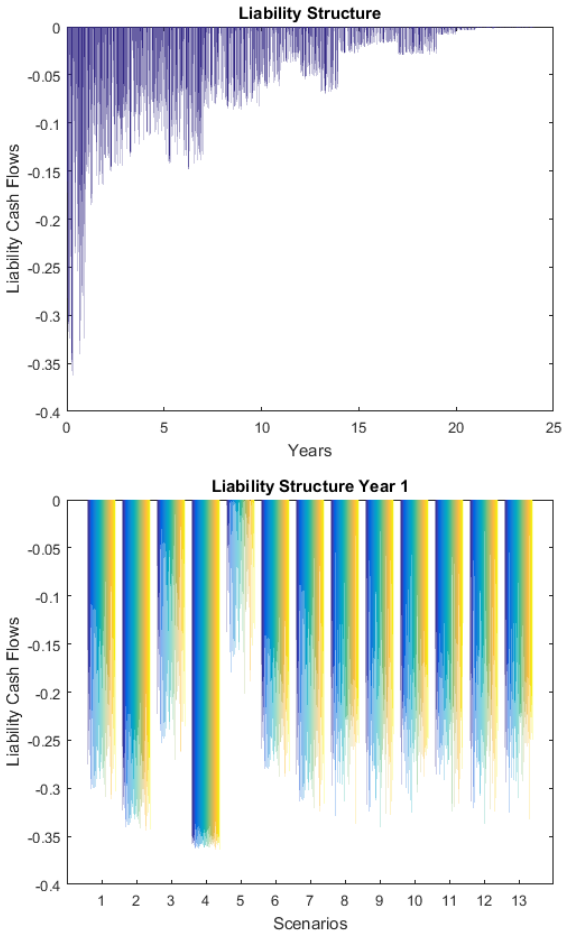
In our example, we use annual cash flows from years 1 to 25 as inputs for the optimization problem. This time window is shorter than the usual time horizon of 40 years and allows us to discuss and visualize the example in a more concise and comprehensible way without losing any informative value.

1.5.2 Structure of the Liabilities

Figure 1.2 shows the structure of our liabilities in general and for the first year of the model in particular. The left plot—showing the discounted liability cash flows over the 25 year time horizon for all scenarios and simulations—indicates that the liabilities clearly decrease over time. This is not a surprise since we are working with discounted cash flows. Furthermore, the cash flows within any given year differ significantly depending on the scenarios and simulations.

The right plot of Figure 1.2 provides more detailed insights regarding the liabilities of the first year. We can see that the liabilities for the base scenario are rather similar to the liabilities of Scenarios 6–13. This observation indicates that changes in the interest rate volatility, the equity index, and the property index do not have a strong impact on the liabilities. On the other hand, the liabilities of Scenarios 2–5—which take changes in the interest rate into account—deviate significantly from the liabilities of the base scenario. In particular, Scenario 4 (a large interest rate decrease) and Scenario 5 (a large interest rate increase) differ strongly from the base scenario. Interest rate decreases (increases)

Figure 1.2: *Liability Structure*



The left plot in this figure shows the liability structure for all thirteen scenarios' 100 simulations over the time horizon of 25 years. The right plot shows the liabilities for year 1 for all thirteen scenarios' and 100 simulations in more detail.

have a negative (positive) impact on Zurich's GAO payments and lead to larger (smaller) liabilities. As a consequence, we particularly need instruments in the RP that are well suited to capture interest rate changes. Aside from Scenario 4, the simulations within each scenario differ significantly from each other. In general the differences between scenarios in the liability structure for years 2–25 are similar to year 1 (with the smaller magnitude indicated in the left plot of Figure 1.2).

1.5.3 Instrument Specification

Table 1.4 describes the set of candidate instruments. We have four different instrument classes: equity indices, cash indices, zero coupon bonds, and receiver swaptions. These instruments have to be able to replicate changes in the liabilities caused by developments of the interest rate and the equity index. They have to be suitable for capturing the growth of a policyholder's account and the annuity payments that he or she receives once the policy matures. The growth of the policyholder's account is mainly based on the equity performance and interest rate developments. Equity indices are suitable for replicating the part of this growth that is based on the performance of the main equity index of a country, while cash indices are suitable for replicating the part of the growth that is based on interest rate developments.

Table 1.4: *Available Instruments*

	Instrument class	Instruments	Notional value
Equity indices	$a = 1$	$I_1 = 25$	$nv_1 = 3000$
Cash indices	$a = 2$	$I_2 = 25$	$nv_2 = 100$
Zero coupon bonds	$a = 3$	$I_3 = 25$	$nv_3 = 100$
Receiver swaptions	$a = 4$	$I_4 = 225$	$nv_4 = 100$

This table presents the available candidate instruments per asset class for the optimization model.

Zero coupon bonds and receivers swaptions are suitable instruments for replicating Zurich's annuity cash flows once a policy matures. Zero coupon bonds replicate fixed payments and are reliable instruments for matching the guaranteed part of the annuity payment.

Receiver swaptions depend on interest rates, and are suitable for replicating the variable part of the annuity payment since they are options on swap rates that benefit from rising interest rates just as the product's payments benefit from higher annuity rates. For a more in-depth analysis of the relationship between swaptions and GAO products, please refer to Pelsser (2003). Finally, we do not need property instruments since the market value of liabilities does not exhibit sensitivity to that risk factor.

1.6 RP Selection Process Results

In this section, we present and interpret the results of a specific RP selection process for our RP implementation example. We first present the previous RP allocation used by Zurich. Second, we discuss the importance of the selection of appropriate candidate instruments. Third, we calculate the unconstrained optimal RP based on all appropriate candidate instruments. Fourth, we add constraints to improve the test performance of the RP. Finally, we evaluate the quality of the optimal RP with the roll forward test.

1.6.1 Previous RP Allocation

When seeking a new optimal RP Zurich always considers the previous RP allocation as a reference benchmark. The third column in Table 1.5 shows the validation and market value test performance for this previous RP. As described in Section 1.5.1, the in-sample scenarios used for the validation test take changes in the interest rate and the equity index—the two factors with the largest impact on Zurich's GAO payments—into account. Additionally, the two out-of-sample scenarios used for the market value test simulate more extreme (and unlikely) scenarios. In Zurich's experience, the previous RP allocation leads to reasonably good test results for almost all scenarios. Here, the worst performances occur in the two scenarios with an increase of the interest rate and in the two out-of-sample scenarios.

Table 1.5: *Validation and Market Value Test Performance*

Test	SN	Scenario feature	Previous	No swaptions	Unconstrained	Optimal
Validation	2	IR small down	−0.84%	−31.46%	−1.06%	−0.33%
Validation	3	IR small up	1.18%	6.22%	1.38%	0.46%
Validation	4	IR large down	0.09%	−66.07%	−0.05%	0.02%
Validation	5	IR large up	1.18%	−12.66%	1.11%	−0.63%
Validation	6	IR vol down	−0.51%	3.49%	−0.47%	−0.34%
Validation	7	IR vol up	0.36%	−3.46%	0.45%	0.34%
Validation	8	EI small down	−0.18%	−3.60%	−0.48%	−0.43%
Validation	9	EI large down	−0.51%	−26.41%	−0.47%	0.08%
Validation	10	EI vol up	0.05%	0.81%	0.03%	−0.02%
Validation	11	EI vol down	−0.02%	−2.86%	0.00%	−0.08%
Validation	12	PI small down	0.00%	−9.87%	0.06%	0.04%
Validation	13	PI large down	0.00%	−2.56%	−0.03%	−0.05%
Validation	-	Average	0.41%	14.12%	0.47%	0.23%
Market value	1	2008 shock	−1.83%	−82.00%	−1.88%	1.03%
Market value	2	Tail shock	−1.28%	−45.21%	−1.78%	0.90%

This table presents the validation and market value test results for the previous RP used by Zurich, the RP with no swaptions, the unconstrained RP, and the RP with the best validation test performance. SN = scenario number, IR = interest rate, EI = equity index, PI = property index, vol = volatility.

1.6.2 Candidate Instrument Selection

In an initial step toward selecting a new RP allocation, we evaluate the appropriate choice of candidate instruments. For our first RP—shown in the fourth column in Table 1.5—we only use equity indices, cash indices, and zero coupon bonds as candidate instruments and no receiver swaptions. For this RP we minimize the L_1 error without any additional constraints. This first step results in an RP with a very poor performance in all scenarios in the validation test. In particular we have very large validation movements for Scenarios 2, 4, 5, and 9.

Notably, this RP shows the worst performance for scenarios with an interest rate decline. The effect clearly increases if the decline in the interest rate gets larger. We have a validation movement of -31.46% for the scenario with a small interest rate decline and a validation movement of -66.07% for the scenario with a larger interest rate decline. This means that the value of the RP drops about two thirds as much as the value of the liability when a small decrease in interest rates occurs, and only about one third as much if a large decrease in interest rates occurs. The RP performs clearly better for the two scenarios with an increase in the interest rate. However, for Scenario 5, which simulates a large increase in the interest rate, the validation movement is also quite large (-12.66%). These results are not surprising and correspond to the function of swaptions as an extremely important instrument for managing interest rate risks (see Pelsser (2003)). The poor performance in the RP without swaptions clearly indicates the importance of adding an instrument class with asymmetric payoffs—an instrument that is able to replicate the behaviour of the liabilities when interest rates decline—to the candidate instruments. Among the instrument classes with asymmetric payoffs that were tested at Zurich, receiver swaptions showed the best results. For the poor performance in Scenario 9, which simulates a large decline in the equity market, we cannot provide such a simple and intuitive explanation. This scenario is one of the most extreme in-sample scenarios and we can only conclude that we need a higher variety of instrument classes to replicate the liabilities in this scenario effectively. The same applies for the two out-of-sample scenarios, which simulate the 2008 shock and a tail shock, respectively.

The RP without swaptions shows terrible market value test results for these scenarios. This is not surprising since these scenarios comprise several economic changes (including interest rate changes) that cannot be replicated appropriately with the small number of given candidate instruments.

In addition, an RP without receiver swaptions leads to undesirable large notional values in cash indices and zero coupon bonds, as shown in Table 1.6. Large notional values usually

Table 1.6: *Notional Values*

Instrument class	No swaptions	Unconstrained	Optimal
Equity indices	34.47%	0.59%	1.17%
Cash indices	1018.51%	38.84%	10.55%
Zero coupon bonds	1105.49%	40.34%	12.09%
Receiver swaptions	0.00%	212.07%	190.00%

This table presents the notional values for our three RPs in percentage of the average liability cash flows per simulation in the base scenario summed up over the entire time horizon and adjusted for the notional value of each instrument class.

reflect an overfitting for the training data set and can also explain the poor performance in Scenario 9, which simulate changes in the equity index.

1.6.3 Unconstrained RP

For the second RP, we add receiver swaptions to our candidate instruments to improve the poor performance observed in the first RP. Receiver swaptions provide a guarantee against decreasing interest rates, as the owner will receive a fixed rate while paying the floating rate. A GAO also protects the policyholder against decreasing interest rates, by guaranteeing a minimum rate for the annuity, which makes receiver swaptions a natural hedge for GAOs. The fifth column in Table 1.5 shows that this addition leads to significantly improved test results. The absolute value of the validation movements for all 12 scenarios is below 1.4 percent, so this RP can capture a wide range of interest rate and equity changes very well. As we can see in Table 1.6, this RP includes large positions in receiver swaptions. In contrast to the first RP, the asymmetric payoff structure of the receiver swaptions ensures that the second RP is able to capture interest rate drops.

This leads to strong validation test improvements for the scenarios that simulate interest rate declines compared to the first RP without swaptions. Additionally, this new RP also replicates the declines in the equity index very well. Consequentially this RP would be well suited to capture Zurich's GAO payments. However, it does not deliver a clear improvement over the previous RP allocation used by Zurich. For several important in-sample scenarios, such as the 1 percent decline and increase in the interest rate, it even has worse test results. Also, the market value tests perform worse for both out-of-sample scenarios.

1.6.4 Optimal RP

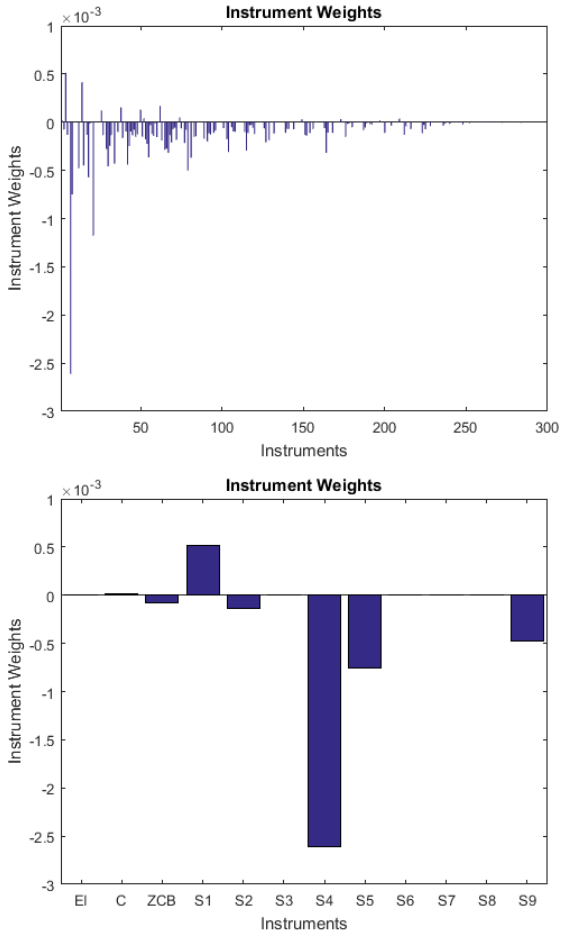
In the final modelling step, we include the constraints (1.11)–(1.13) to build the full linear programming model. In total, we solve this LP for 656 different parameter combinations and so obtain 656 different RPs. For constraint (1.11) we use 16 different notional limits, nl, between 1 and 2.5 in steps of 0.1. We combine each possible value for nl with 41 different values for $adub = -adlb$ (AAD) between 0.001 and 0.005 in steps of 0.0001. For the 656 RPs, we compare the average of the absolute validation test result values for all 12 in-sample scenarios. Out of all these portfolios, the RP with $AAD = 0.0022$ and $nl = 1.9$—which we refer to as the "optimal portfolio" in the remaining discussion—performs best with an average validation test of 0.23 percent. This result is clearly better than the average validation test value of the RP without swaptions (14.12%), the unconstrained RP (0.47%), and also the previous RP allocation (0.41%). Additionally, the detailed validation test results for each scenario of the optimal RP shown in the sixth column in Table 1.5 indicate that this RP is significantly better suited to capturing declines in the interest rate than are the other RPs. Since declining interest rates have a particularly strong impact on Zurich's GAO payments, the test performances in Scenarios 2 and 4 are highly relevant and the excellent test results of the optimal RP in these particular scenarios are a strong argument for its adoption.

Since we selected the optimal RP based on the validation test results, the excellent performance in this test is of little surprise. To confirm these excellent results, we also

evaluate the market value test performance for the two out-of-sample scenarios of the optimal RP. For both shocks, the optimal RP clearly outperforms the two other RPs and the previous allocation. A last argument for this choice of optimal RP are its notional values. For all instrument classes (apart from the equity indices) the notional values are significantly smaller than for the unconstrained RP. This feature minimizes the overfitting for the training data set and allows for the excellent validation and market value test results. The receiver swaptions are the only instrument class to hit the upper position limit. This is no surprise since receiver swaptions are the largest instrument class among our candidate instruments. Furthermore, receiver swaptions are the best suited of all available instrument classes to replicating the strong liability changes induced by increases and decreases in the interest rate, as described in Section 1.5.3.

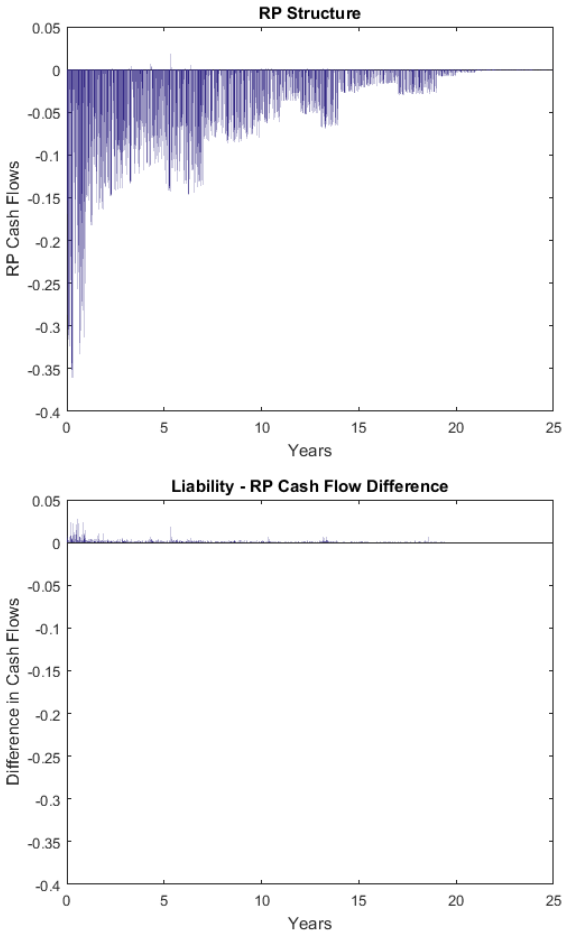
Figure 1.3 shows the instrument weights for our optimal portfolio. The left plot provides an overview for all instruments, while the right plot depicts the weights of the instruments that have cash flows in the first year in more detail. We have holdings in each of the 300 candidate instruments, but—as shown in the right plot—for a significant number of instruments the weights are close to zero. These instruments with a weight close to zero are still used in practice but they can be ignored for the interpretation of the portfolio. Therefore our RP can still be considered as a "small RP" that can be well interpreted in relation to the liabilities. The small cash holdings and very small equity index holdings (which are clearly visible in the right plot and also appear in the left plot) indicate that changes in the growth of the policyholders accounts do not have a large impact on the liabilities and do not need to be replicated with equity indices or cash. On the contrary, we have large holdings in receiver swaptions that can replicate the significant effect of interest changes on the liabilities. The left plot in Figure 1.4 shows the structure of the RP cash flows. Evidently this plot replicates the liability structure—shown in the left plot of Figure 1.2—very well. The extremely small difference between liability and RP cash flows for the optimal portfolio is shown in the right plot of Figure 1.4. This confirms how well our RP replicates our liabilities and that we do not have a single simulation with a very poor replication result. Furthermore, the RP only includes a negligible amount of

Figure 1.3: *Optimal RP—Instrument Weights*



The left plot in this figure shows the weights for all 300 instruments. The right plot shows the liabilities for the 12 instruments with cash flows in year 1 in more detail. EI = equity index, C = cash, ZCB = zero coupon bonds, S = receiver swaptions.

Figure 1.4: *Optimal RP—Cash Flow Structure*



The left plot in this figure shows the RP cash flow structure for all thirteen scenarios' 100 simulations over the time horizon of 25 years. The right plot shows the difference between RP and liability cash flows.

positions with positive cash flows (which proofs the success of Constraint (1.11)). These very few positions with positive cash flows in the RP are not systematic and are therefore considered as "noise" (and not interpreted).

1.6.5 Final RP Evaluation

As discussed in Section 1.4, Zurich's main success test is the roll forward quality metric, as described in equation (1.26).

First, we identify $\widetilde{X}$, the average tail shocks. In this example, we conduct the roll forward test assuming $\widetilde{X}$ includes an interest rate downward movement of -1.5 percent and an equity index downward movement of -40 percent. The actual values are not as important as the concept of what $\widetilde{X}$ represents, since the actual values will be different for every product, for every company and will also change over time in response to market conditions.

In our RP implementation example, GAO payments are the main concern. Therefore, scenarios simulating interest rate and equity index declines have the highest impact for us and we will limit the example to these two dimensions. We could have added other dimensions p for $\widetilde{X}_p$, but they would not affect the results materially.

Second, we need to identify $x_p^u = X_p^{\mathcal{K}(p)}$ and $x_p^l = X_p^{\mathcal{N}(p)}$. Here again, the actual values are not as important as the concept. All the sensitivities $X_p^{\mathcal{K}(p)}$ must be in the tail and $X_p^{\mathcal{N}(p)}$ must be chosen such that $\widetilde{X}_p$ lies between $X_p^{\mathcal{K}(p)}$ and $X_p^{\mathcal{N}(p)}$. The example has been constructed in such a way that these are equidistant from $\widetilde{X}_p$. This means that $\frac{x_u - \widetilde{X}_p}{x_u - x_l} = \frac{\widetilde{X}_p - x_l}{x_u - x_l} = 0.5$

For the interest rate we calculate the errors for Scenario 2 (small interest rate down) and Scenario 4 (large interest rate down) based on K^{NE}, and embrace the linearity assumption

to find the error for the interest rate decline of -1.5 percent:

$$R_{IR:-1.5\%} = \left[0.5\left((RL_{k_2^{NE}} - RL_{k_1^{NE}}) - \sum_{a=1}^{A}\sum_{i=1}^{I_a}(RI_{k_2^{NE},a,i}(w_{a,i}) - RI_{k_1^{NE},a,i}(w_{a,i}))\right)\right.$$
$$\left.+0.5\left((RL_{k_4^{NE}} - RL_{k_1^{NE}}) - \sum_{a=1}^{A}\sum_{i=1}^{I_a}(RI_{k_4^{NE},a,i}(w_{a,i}) - RI_{k_1^{NE},a,i}(w_{a,i}))\right)\right] / \sum_{a=1}^{A}\sum_{i=1}^{I_a} RI_{1,a,i}(w_{a,i}). \tag{1.27}$$

We follow a similar approach for the equity index. We calculate the errors for Scenario 8 (equity index small down) and Scenario 9 (equity index large down) based on K^{NE}, and embrace the linearity assumption to find the error for the equity index decline of -40 percent:

$$R_{EI:-40\%} = \left[0.4\left((RL_{k_8^{NE}} - RL_{k_1^{NE}}) - \sum_{a=1}^{A}\sum_{i=1}^{I_a}(RI_{k_8^{NE},a,i}(w_{a,i}) - RI_{k_1^{NE},a,i}(w_{a,i}))\right)\right.$$
$$\left.+0.6\left((RL_{k_9^{NE}} - RL_{k_1^{NE}}) - \sum_{a=1}^{A}\sum_{i=1}^{I_a}(RI_{k_9^{NE},a,i}(w_{a,i}) - RI_{k_1^{NE},a,i}(w_{a,i}))\right)\right] / \sum_{a=1}^{A}\sum_{i=1}^{I_a} RI_{1,a,i}(w_{a,i}). \tag{1.28}$$

Table 1.7 shows the results for the roll forward test for our optimal RP and for the RP without swaptions and the unconstrained RP, which serve as a reference. The roll forward test performance for the equity decline is excellent and the performance for the interest rate decline is good. Taken as a whole, the optimal RP has significantly better

Table 1.7: *Roll Forward Test Results*

Sensitivity	No swaptions	Unconstrained	Optimal
IR	30.70%	1.68%	1.89%
EQ	11.63%	0.24%	−0.01%
Total	42.43%	1.92%	1.91%

This table presents the roll forward test results for our three RPs.

results than the RP without swaptions and slightly better ones than the unconstrained RP. These test results mean that the optimal RP implies an error in economic capital of

approximately 2 percent. This measurement takes place at the same time as the economic capital calculations (6 months after the original calibration), so the quality is being measured together with the underlying metric of interest. This makes the measurement more relevant to the purpose of the calculation.

1.7 Impact of the RP Model at Zurich

In the previous sections we described the model as implemented at Zurich and provided one numerical example of this model using real-life data. In this section, we describe the use of the model to provide the context for the calculations shown in Section 1.6. We show that the model is an important part of risk management efforts at Zurich, serving two major purposes for the company: It makes economic capital calculations possible by enabling fast calculations of liability values and it contributes to a sound understanding of Zurich's liabilities. All this is done in the context of the strict regulatory requirements for insurance companies.

1.7.1 Risk Management at Zurich

Within Zurich's risk management strategy, one of the company's risk management objectives is to protect its capital base. To achieve this objective, Zurich closely monitors different types of risk to ensure that it does not take risks that exceed its own risk tolerance.

Zurich developed its "Economic Capital Model" (Z-ECM) to assess its business's economic capital consumption. The RP model is situated at the core of the Z-ECM model and is used to represent the life liabilities in the market risk sub-module. Market risk is the risk associated with the balance sheet positions where value or cash flows depend on financial markets. Risk factors include equity market prices, property market prices, interest rates, credit spreads, and currency exchange rates (see Zurich Insurance Group (2017)). In the short term, market risk is primarily driven by economic and financial conditions, which can change from year to year. Various scenarios for these risk factors are included in the scenarios that determine the parameter values for the RP optimization

model, as we show in the example in Section 1.5.

Once the RPs are calculated for all material business lines, they are used as inputs for the market risk model. Results for the Z-ECM and the Swiss Solvency Test are taken from this model. Without a proxy model such as RPs, Zurich would not be able to calculate its Z-ECM or fulfill its regulatory requirements with regard to the Swiss Solvency Test. Additionally, as RPs are portfolios of financial instruments, they are easier to integrate within the investment management framework, allowing the company to align risk taking and risk measurement. In this regard, RPs give Zurich an advantage in terms of consistency within the risk framework. This fact is explored in more depth in the following section.

1.7.2 Understanding Liabilities

Besides offering fast calculation times, the RP model is also well suited for understanding the liabilities in the balance sheet. Zurich states in its 2016 annual report that "the group manages the market risk of assets relative to liabilities on an economic, total balance sheet basis"(Zurich Insurance Group (2017)). The foundation for managing Zurich's investment position is its understanding of the liabilities it has accepted onto its balance sheet. RPs are fundamental to achieving this objective as they allow the formation of a "total balance sheet" perspective at group level—that is to say, a consolidated view that results from aggregating all the individual business units in the insurance group. This can be understood by thinking about the replication process as a translation exercise that takes cash flows as inputs and returns financial instruments, thus "translating" liabilities into a common language shared with the other side of the balance sheet—the instruments. This translation also helps with the problem faced by every company with global operations: the aggregation of information from very diverse countries with possibly different currencies into a single, consolidated, group view.

The way in which this is achieved can be understood by examining an overview of the information flow. As each country or business unit within Zurich underwrites insurance business, they also build models to calculate both the reserves necessary to support

that business and the value of that business to Zurich's shareholders. These models are based on projecting the expected cash flows of the policies concerned. These expected cash flows are then used to calibrate the RPs for that business. Once this process is complete, the RPs are passed on to the market risk model, where they form the basis of the simulated balance sheet underlying the economic capital calculations. Additionally, that same model provides the necessary inputs for strategic asset allocation in the company. In this regard, Zurich is one of the few companies that embeds its RP model not only in its economic capital model but also in its investment management framework.

In summary, RPs allow a consistent liability information flow across the company—which is always a hard challenge in a large organization—and thus improve the decision-making process at different levels. Even though this factor cannot be quantified in the same way as can the quality or the speed of the process, it is of crucial importance to the company.

1.8 Conclusion

In this paper, we have analyzed the challenge that a life insurance company faces with regard to economic capital calculations. We have developed an RP model to solve this problem using the available amount of data and time. The model minimizes the L_1 error between an insurance company's discounted liability cash flows and the discounted RP cash flows. We have also explained the generation of the data samples that contain the necessary scenarios and simulations for our model. In addition, we have introduced several tests for an ex post evaluation of the optimal solution to the RP model.

We implemented the model using real-life data sets and have demonstrated that our approach performs well on large-scale real-life data. In our implementation example we explained the essential economic conditions that we considered for our scenarios. And we explained the process of finding an optimal RP. First, we evaluated the choice of appropriate candidate instruments. Second, we imposed constraints to optimize our validation test results. Third, we confirmed the good validation test performance of our optimal RP with excellent market value test results and interpreted the RP in relation to the liabilities. In a final step, Zurich's main success test—the roll forward metric—

confirmed the high quality of our RP.

Finally, we explained how our RP optimization model contributes to risk management at a large global insurance company. The RP model directly enters the calculations of the company's economic capital requirements. Thereby, it has the advantage that it is easier to integrate within the investment management framework compared to other proxy models.

To the best of our knowledge, this paper presents the first extensive analysis in the operations research literature of the actual implementation of an RP optimization model for the life insurance industry. The analysis in this paper shows that questions relating to RPs can certainly be a source of stimulating and highly relevant applied work in the field of operations research.

Appendix

1.9 Alternative Linear Formulation

An alternative linear reformulation—see Konno and Yamazaki (1991) for an application to equity portfolios—has been previously suggested for RPs; see Footnote 1 in Chen and Skoglund (2012a). For this reformulation, we introduce a set of auxiliary variables, $z_{k^{IS},c^{TR},t}$, and minimize their sum in the new linear objective function

$$min_{l,s,z} \sum_{k^{IS}=1}^{K^{IS}} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} z_{k^{IS},c^{TR},t}. \tag{1.29}$$

In addition, we need to introduce constraints relating each auxiliary variable to a term in the original objective function in the L_1 problem (1.5),

$$z_{k^{IS},c^{TR},t} \geq LCF_{k^{IS},c^{TR},t} - \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{k^{IS},c^{TR},t,a,i}(l_{a,i} - s_{a,i}) \qquad \forall\ k^{IS}, c^{TR}, t \tag{1.30}$$

$$-z_{k^{IS},c^{TR},t} \leq LCF_{k^{IS},c^{TR},t} - \sum_{a=1}^{A} \sum_{i=1}^{I_a} ICF_{k^{IS},c^{TR},t,a,i}(l_{a,i} - s_{a,i}) \qquad \forall\ k^{IS}, c^{TR}, t \tag{1.31}$$

Numerical tests show that this formulation leads to significantly longer running times than the formulation of Feinstein and Thapa (1993) that we used in our paper.

1.10 Regularization

It is possible to add a regularization term to equation (1.5):

$$f(w) = \sum_{k^{IS}=1}^{K^{IS}} \sum_{c^{TR}=1}^{C^{TR}} \sum_{t=1}^{T} \left| LCF_{k^{IS},c^{TR},t} - \sum_{a=1}^{A} \sum_{i=1}^{I_a} w_{a,i} ICF_{k^{IS},c^{TR},t,a,i} \right| + \sum_{a=1}^{A} \sum_{i=1}^{I_a} \lambda_{a,i} \left| w_{a,i} \right|, \tag{1.32}$$

where $\lambda_{a,i}$ is a regularization factor that is set to equal the absolute value of the base value of an instrument multiplied with a parameter that is used to "discourage" the use of certain instruments. Considering the existence of highly correlated market variables in the simulations, adding some type of regularization is good practice in terms of preventing overfitting. The regularization term is commonly based on L_1 or L_2 norms. We use the

former in this model. Both norms improve the out-of-sample performance by shrinking large regression coefficients in order to reduce overfitting, but the L_1 norm additionally leads to less non-zero coefficients, which results in a more parsimonious model.

The presence of highly correlated instruments in the instrument universe means that it is not possible to deduce from the data which instruments should be used. While the LASSO method as implemented yields satisfactory results, it is somewhat indifferent to the choice among a set of strong but correlated variables (Friedman et al. (2001)). Exploring other regularization methods such as Group Lasso or Elastic Net is a task for future research.

An extremely important step for improving the out-of-sample performance is to limit extreme positions in the portfolio composition. Any set of scenarios used for the calibration can include implicit assumptions of high correlation between different market variables that might not hold under a different set of scenarios. Extreme derivative positions that help to achieve a good in-sample fit by offsetting each other can lead to an unstable balance that might break under a different set of scenarios. A real-world example of such an occurrence is the case of Long-Term Capital Management, a hedge fund that famously had to be bailed out in 1998 with an injection of USD 3.6 billion (see Jorion (2000)) after large, extremely leveraged derivatives positions calculated by a model to exploit minuscule arbitrage opportunities led to large losses following an un-modelled shock. In this example the Russian debt restructuring of 1998 is an out-of-sample scenario in which historical correlations and volatilities are no longer reliable.

1.11 Trimming

Theoretically, our two optimization problems may have multiple optimal solutions. In particular, there may be multiple solutions with different values for $l_{a,i}$ and $s_{a,i}$ for some a, i, but which are otherwise identical. Such a portfolio has unnecessarily offsetting long and short positions in the same asset. Obviously, the more suitable solution has the feature that either $l_{a,i}$ or $s_{a,i}$ is zero. Therefore, we need to calculate the trimmed solution for every feasible solution (z, l, s). In this trimmed solution, $l_{a,i}s_{a,i} = 0$ for $i = 1, \ldots, I_a$

and $a = 1, \ldots, A$ (see Jacobs et al. (2005)). According to the description of trimming used in Jacobs et al. (2006), we remove the offset from simultaneous long and short positions in our candidate instruments in such a way that the smaller of these positions diminishes to zero. This trimmed solution $(z, \bar{l}, \bar{s})$ is defined as follows:

$$\bar{l}_{a,i} = l_{a,i} - \delta_{a,i}, \quad \bar{s}_{a,i} = s_{a,i} - \delta_{a,i}, \qquad \text{for all } i = 1, \ldots, I_a, a = 1, \ldots, A, \tag{1.33}$$

where $\delta_{a,i}$ is given by

$$\delta_{a,i} = min\{l_{a,i}, s_{a,i}\} \qquad \text{for all } i = 1, \ldots, I_a, a = 1, \ldots, A. \tag{1.34}$$

Trimming an optimal solution of the LP retains optimality.

1.12 Possible improvements to Roll Forward Test

The approximation of the expected shortfall error described above relies on several assumptions, caused by the nature of the problem and the available data. The two main assumptions are $\mathcal{T}_{A-RP} = \mathcal{T}_{A-L}$ (order preservation) and the linearity of $\mathcal{E}(X)$. The first can be checked (and has been) with the K^{NE} data set. The second most certainly does not hold for the entire domain of $\mathcal{E}(X)$, but it might not be an unreasonable approximation for smaller intervals. A possible improvement on the locally linear assumption is to perform a piecewise linear interpolation of $\mathcal{E}(X)$, which would improve accuracy by better utilizing all the information in K^{NE} and could deal with the problematic case of $A(X) - RP(X)$ being concave, that is, having tail scenarios in both ends of $X_{(i)}$ for a particular i.

An important decision in the construction of this approximation is the treatment of interest rate yield curves. We assume that there is one variable $X_{(i)}$ per relevant spot rate (which means per yearly step in practice). This can mean that there are at least forty interest rate variables. However, it is neither practical nor necessary to require one sensitivity for each of these variables. The simplest way to reduce the problem dimensionality is parallel movements. For a more accurate modelling of this risk factor, it is possible to

use principal component analysis to extract a small number of the most relevant movements (two to six is a good number in practice).

2 A neural network model for solvency calculations in life insurance

Lucio Fernandez-Arjona, University of Zurich

Abstract

Insurance companies make extensive use of Monte Carlo simulations in their capital and solvency models. To overcome the computational problems associated with Monte Carlo simulations, most large life insurance companies use proxy models such as replicating portfolios.

In this paper, we present an example based on a variable annuity guarantee, showing the main challenges faced by practitioners in the construction of replicating portfolios: the feature engineering step and subsequent basis function selection problem.

We describe how neural networks can be used as a proxy model and how to apply risk-neutral pricing on a neural network to integrate such a model into a market risk framework. The proposed model naturally solves the feature engineering and feature selection problems of replicating portfolios.

2.1 Introduction

Insurance companies rely on financial models for quantitative risk management. These risk models should be accurate and fast in terms of the calculation of risk figures such that the rapid pace of market environments is matched. Life insurance companies face the challenge of having to quickly revalue their liabilities under economic stress scenarios based on market-consistent valuation principles.

This paper is based on our presentation at the 2019 Insurance Data Science Conference (ETH Zurich). We thank the scientific committee for its invitation to present, and all those that provided helpful comments during the preparation: Mariana Andres, Dr Gregor Reich, and the participants of the Quantitative Business Administration PhD Seminar. Special thanks go to Prof. Karl Schmedders and Prof. Damir Filipovic, for reviewing the presentation and for all we have learnt from them, without which this paper would not be possible.

Typically, insurance liabilities exhibit features, such as options and guarantees, comparable to standard financial products. Unlike for the latter, there are generally no closed-form formulas for the valuation of the former. Because of this, the use of numerical methods, such as Monte Carlo techniques, becomes inevitable. However, the choice of the specific technique to be used is the key factor in the accuracy and speed of the calculation of risk figures.

Nested Monte Carlo, a straightforward approach to this problem, is computationally burdensome to the point of being infeasible in most cases. Other standard techniques to approach this problem are the least squares Monte Carlo approach (LSMC) and the replicating portfolio approach (RP). These techniques, and their advantages and disadvantages, will be described in more detail in the following sections.

The availability of accurate and fast valuation methods is of great interest to risk management practitioners in life insurance companies. In the last decade, machine learning models based on neural networks have emerged as the most accurate in many fields, among them image recognition, natural language understanding and robotics. While not the first to apply neural networks to life insurance modelling, this paper incorporates the idea of risk-neutral valuation of neural networks to achieve higher accuracy than other existing models while remaining within a realistic computational budget.

A review of the literature shows that several papers work with machine learning techniques to address the problem of calculating the value and risk metrics of complex life insurance products.

One family of methods starts with exact information about a small number of contracts, and then uses spatial interpolation to generate valuations for all contracts. For example, Gan and Lin (2015) use a clustering technique for the selection of the representative contracts and then apply a functional data analysis technique, universal kriging, for the interpolation. Hejazi and Jackson (2016) propose using a neural network model for the interpolation step instead of traditional interpolation techniques. These methods interpolate among policies for a fixed set of economic scenarios, hence reducing the computational burden of calculating the value of the insurance contract. They address the valuation

problem but on their own they are not enough to solve the computational problem of calculating risk metrics.

The effective calculation of risk metrics is addressed by a family of methods that take portfolio valuations (or cash flows) as inputs and use regression methods to construct a model capable of producing the real-world distribution of values of the insurance portfolio. These methods can be classified in two groups, *regress-now* and *regress-later*, a classification first proposed by Glasserman and Yu (2002). Regress-later methods perform better than regress-now methods, as shown by Beutner et al. (2013). However, regress-later methods require setting larger optimization problems, which leads to regress-now methods being computationally cheaper.

Examples of both types of methods in this family can be found in Beutner et al. (2016) and Castellani et al. (2018). The former presents a regress-later model based on an orthogonal basis of piece-wise linear functions, and the latter a regress-now model based on neural networks. LSMC and replicating portfolios belong to this overarching family of methods, and any machine learning approach in this family can be a direct replacement for them.

The models in Beutner et al. (2016) and Castellani et al. (2018) are far from the only examples of regression methods applied to the problem of solvency capital calculation. Many people have worked on this problem and made contributions to the field. A regress-now model based on Gaussian process regression was presented in Risk and Ludkovski (2018), and many more models can be found cited in the introduction to Beutner et al. (2016). In this introduction we highlight only those we see as closest to our work.

In this paper we present a regress-later model based on neural networks, including a closed-form formula for its risk-neutral valuation, and compare it to a benchmark implementation of replicating portfolios. We focus on this comparison in order to answer a question of relevance to practitioners: *Can neural networks provide better results than existing methods for solvency capital calculations used in the industry?* A formal mathematical treatment of the methods involved can be found in the already mentioned Beutner et al. (2013) (regress-later vs regress-now methods), Natolski and Werner (2014b), and

Cambou and Filipović (2018) (RPs) and Bauer et al. (2010) (LSMC).

We have not found in the literature previous work on comparing replicating portfolios (that is, regress-later method with financial instruments as basis functions) to other approximation techniques. There are, however, several papers that compare LSMC approaches: Bauer et al. (2010) present a comparison between LSMC and nested Monte Carlo, and Pelsser and Schweizer (2016) present a comparison between LSMC regress-now and LSMC regress-later.

Against the background described above, this paper's contribution is threefold:

- it presents a reproducible replicating portfolio approach suitable for benchmarking in a research context,
- it introduces a risk-neutral valuation formula for a class of neural networks with multivariate normally distributed inputs (such as discrete time Brownian motion processes),
- it builds a regress-later methodology based on neural networks and compares the quality of the economic capital calculations between this method and the replicating portfolio approach.

The neural network model improves on those in use in the industry and some of those presented in the literature. In comparison with the (regress-now) neural network model in Castellani et al. (2018), our model is based on a regress-later approach, which—as described in the literature—provides more accurate results than regress-now models. In comparison with the regress-later approach in Beutner et al. (2016), the neural network approach allows us to avoid having to define an arbitrary dimensionality reduction function, and a hypercube grid. Under a neural network model, those parameters are data driven and determined as part of the optimization.

The rest of the paper is structured as follows: Section 2.2 describes the solvency capital calculation problem in detail, together with possible solutions based on nested Monte Carlo and proxy models. Section 2.3 summarizes the mathematical framework common to all regression models (both regress-now and regress-later), Section 2.4 describes the

proposed neural network approach, and presents the risk-neutral valuation of a class of neural networks. Section 2.5 discusses important qualitative aspects of the models, such as model complexity and feature engineering. Finally, Section 2.6 describes the numerical experiments and presents the results.

2.2 Solvency capital calculation problem in life insurance

Solvency regimes—including Solvency II or Swiss Solvency Test—require companies to hold capital in excess of a legal minimum that is based on the amount necessary to remain solvent with high confidence in a one-year period.

The above implies the determination of the distribution of the value of the asset–liability portfolio at the end of the one-year period. We call the value of the asset–liability portfolio V_t, and therefore V_0 and V_1 are the initial value and the value at the end of the first year, respectively. The solvency capital requirement at confidence level α is then determined relative to a risk metric applied to the distribution of $\Delta V = V_1 - V_0$, where V_0 is considered a constant:

- Value at risk (Solvency II)

$$VaR_\alpha(\Delta V) = -\inf\left\{v : F_{\Delta V}(v) > 1-\alpha\right\} = -F^{-1}_{\Delta V}(1-\alpha).$$

- Expected shortfall (Swiss Solvency Test)

$$ES_\alpha(\Delta V) = \frac{1}{1-\alpha}\int_\alpha^1 \mathrm{VaR}_\gamma(\Delta V)d\gamma.$$

If F_V^{-1} is not known (and this is usually the case), then we must simulate $\{V_1^{(i)}\}_{i=1:M}$ and then calculate the risk metric on the empirical (simulated) distribution. These M samples are referred to as real-world (or *natural*) simulations.

Having described how risk calculations usually depend on a Monte Carlo sampling of V_1, $\{V_1^{(i)}\}_{i=1:M}$, we now focus on the calculation of each individual $V_1^{(i)}$. These represent the value of the asset–liability portfolio at the end of the one-year period. The valuation must

be carried out on a market-consistent basis, which means applying risk-neutral valuation:

$$V_t = E_t^{\mathbb{Q}}\big[\sum_{\tau>t} CF_\tau(X)\big]. \tag{2.1}$$

In the formula above the value of the portfolio is the expectation over the risk-neutral measure $\mathbb{Q}$ of future discounted cash flows, $CF_\tau(X)$. These cash flows depend on a set of economic variables X. In turn, X depends on a smaller set of normally distributed random drivers ξ—that is, $X = X(\xi)$. This implies that there is a $CF'_\tau(\xi)$ such that $CF' = CF \circ X$.

When calculating V_1 for solvency capital purposes, the real-world simulations determine an empirical distribution of $X_{0:1}$ (X between 0 and 1), and for each sample in such distribution there is a risk-neutral distribution of X after $t = 1$—which we call $X_{1:T}$—over which V_1 must be calculated.

For simple products, such as a standard annuity, the calculation of V_1 is straightforward. However, for products that lack a closed form formula, $\{V_1^{(i)}\}_{i=1:M}$ must be approximated by some $\{\widehat{V}_1^{(i)}\}_{i=1:M}$. Most complex products, such as variable annuities, fall under this case.

2.2.1 Nested Monte Carlo

A simple solution for approximating V_1 is to apply a Monte Carlo approach:

$$\widehat{V}_t = V_{MC} = \frac{1}{N}\sum_{j=1}^{N}\sum_{\tau>t} CF_\tau(X_{t:T}^{(j)}|X_{0:t}).$$

This formula implies taking N samples and calculating the average of the discounted simulated cash flows. Since this must be done for each real-world simulation i to be able to construct $\{V_1^{(i)}\}_{i=1:M}$, the full simulated distribution is given by

$$\{\widehat{V}_t^{(i)}\}_{i=1:M} = \Big\{\frac{1}{N}\sum_{j=1}^{N}\sum_{\tau>t} CF_\tau(X_{t:T}^{(j)}|X_{0:t}^{(i)})\Big\}_{i=1:M}.$$

The set of risk-neutral scenarios are called the inner scenarios because they are con-

structed for each outer scenario i to estimate the value of the portfolio conditional on the information at time 1. Since a total of $M \times N$ simulations are required, a nested Monte Carlo approach is usually infeasible. Most insurers, therefore, use other approximation methods for $\widehat{V}_1$, the most popular being *least squares Monte Carlo* (LSMC) and *replicating portfolios* (RPs). Both these methods are based on a regression approach. For an analysis and comparison of nested Monte Carlo to regression-based methods, the reader is referred to Broadie et al. (2015).

2.2.2 Alternatives to nested Monte Carlo

Given the computational difficulties of nested Monte Carlo, many methods have been proposed in the literature, some of which are in place in the industry. With regard to those in place, it is important to note that none of these proxy models replace the original, full insurance cash flow model of the asset–liability portfolio. These methods focus on allowing the solvency capital to be calculated with fewer executions of that model.

LSMC, originally introduced for pricing American style derivatives by Longstaff and Schwartz (2001), is used to reduce the number of necessary risk-neutral simulations by finding a polynomial approximation of the portfolio value as a function of the risk drivers. The coefficients of the polynomial expansion are obtained from a regression against a reduced-size nested Monte Carlo. LSMC is usually applied in the industry as, in the terminology of Glasserman and Yu (2002), a *regress-now* approach, as described in Bauer et al. (2010). However, polynomial approximations do not need to be restricted to *regress-now* applications.

The replicating portfolio approach is based on running a regression against the portfolio cash flows, but instead of polynomials the model uses a set of financial securities as basis functions. The problem is formulated as a linear optimization problem (L_1 or L_2) where the objective is to minimize the differences between the cash flows of the asset–liability portfolio and the replicating portfolio. The output is a portfolio of financial instruments that reproduces the payout of the life insurance portfolio as closely as possible. For reference, Natolski and Werner (2014b) analyse in detail some of the popular approaches

to constructing replicating portfolios. Vidal and Daul (2009) and Chen and Skoglund (2012b) look at replicating portfolios from a more practical point of view and Adelmann et al. (2019) provide a description of a real-world implementation in the insurance industry.

In this paper we present a method based on neural networks that combines the strengths of LSMC and replicating portfolios while providing higher accuracy in a setting with a realistic amount of inputs and computing power. We stress this last point since it is common for studies on neural networks to provide results based on millions of input samples, which would not be practical in the real world.

2.3 Mathematical formulation

Both regress-now and regress-later models are estimators for the value of the asset–liability portfolio, $\widehat{V}_t$. They differ in the input variables that they use and the target variable that they estimate.

Regarding the input variables, regress-now models work with the input economic variables until time t, while regress-later use all variables until the end of the projection horizon. Regarding the target variable, regress-now models directly estimate the time-t value function while regress-later models estimate the discounted cash flows —individual, grouped or terminal.

Regress-now models approximate the value function by estimating the coefficients $\{w_k\}$ in

$$\widehat{V}_t^{(i)}(X) = \sum_k w_k \phi_k(X_{0:t}^{(i)}).$$

This estimation is done via regression, minimizing the squared error

$$\sum_{i=1}^{m} \Big(\sum_k w_k \phi_k(X_{0:t}^{(i)}) - \tilde{V}_{MC}^{(i)} \Big)^2,$$

where $\tilde{V}_{MC}^{(i)}$ differs from $V_{MC}^{(i)}$ in that it is calculated with a very low number n of inner simulations, instead of using N simulations as in nested Monte Carlo. It is also worth noting that the regression is performed over $m << M$ outer scenarios. LSMC uses a

polynomial basis for $\{\phi_k\}$. Once the model has been calibrated, it can be used to *predict* (in its machine learning sense) all M scenarios, which were not part of the training data. Regress-later models approximate the value function by estimating the coefficients $\{w_k\}$ necessary to build the following estimator:

$$\begin{aligned}\widehat{V}_t^{(i)} &= E_t^{\mathbb{Q}}\big[\sum_{\tau>t}\widehat{CF}_\tau(X)\big] \\ &= E_t^{\mathbb{Q}}\big[\sum_{\tau>t}\sum_k w_{k,\tau}\phi_{k,\tau}(X_{0:\tau})\big] \\ &= \sum_{\tau>t}\sum_k w_{k,\tau}E_t^{\mathbb{Q}}\big[\phi_{k,\tau}(X^{(i)})\big],\end{aligned} \tag{2.2}$$

where the variable τ indicates that a different set of basis functions $\phi_{k,\tau}$ and parameters $w_{k,\tau}$ is used for each timestep $\tau > t$.

The equation above shows that, while more accurate than regress-now models, regress-later models impose an additional requirement: to be able to calculate $E_t^{\mathbb{Q}}[\phi_{k,\tau}]$. In the best case, there is a closed-form formula. In the worst case, it can be done via Monte Carlo, but the computational cost will partially or completely offset the computation gains of avoiding nested Monte Carlo on the full model.

This estimation is done via regression, minimizing the error

$$\sum_{i=1}^{m}\sum_{j=1}^{n}\sum_{\tau>t}\Big|\sum_k w_{k,\tau}\phi_{k,\tau}(X_{t:T}^{(j)}|X_{0:t}^{(i)})) - CF_\tau(X_{t:T}^{(j)}|X_{0:t}^{(i)})\Big|^p,$$

where (as in the LSMC case) $m << M$ but (unlike in that case) n could be as large as N if required.

Typically this regression is done minimizing squared errors ($p = 2$) but some companies use absolute errors ($p = 1$). The approximation is based on a linear regression at each τ of $CF_\tau(\cdot)$ against $\{\phi_{k,\tau}(\cdot)\}$, the cash functions of a set of financial instruments (bonds, swaps, equity options). This is the usual case in the industry, although some older implementations used grouped cash flows (in time buckets) and some papers, like Cambou and Filipović (2018), present the topic in terms of terminal values (that is, all time steps

grouped).

The neural network approach proposed in this paper will take the form of a regress-later method, but using neural network structure for basis functions and performing a non-linear optimization to find its parameters.

2.4 A neural network approach

Artificial neural networks, more commonly referred to as neural networks, constitute a broad class of models. Among them, the simplest is the single-layer perceptron, which we use for our model.

The single-layer perceptron, a type of feed-forward network, is a collection of connected units or nodes arranged in layers. Nodes of one layer connect only to nodes of the immediately preceding and immediately following layers. They are fully connected, with every node in one layer connecting to every node in the next layer. The layer that receives external data is the input layer. The layer that produces the ultimate result is the output layer. In between there is one hidden layer. Each node is a simple non-linear function $\phi(\cdot)$ applied to a linear combination of its inputs—that is, those nodes to which is connected. An example of this architecture is shown in Figure 2.1, for three inputs and a hidden layer with a width of four nodes.

The single-layer perceptron is a universal function approximator, as proven by the universal approximation theorem (Hornik (1991)).

The first neural network model that we present is a direct equivalent of Equation (2.2) for the case of single-layer perceptron:

$$\widehat{V}_t = E_t^{\mathbb{Q}}\big[\sum_{\tau>t}\sum_{k} w_{k,\tau}\phi_{k,\tau}(X_{0:\tau})\big] = E_t^{\mathbb{Q}}\big[\sum_{\tau>t}\sum_{k} w_{k,\tau}\phi(\mathbf{v}_{k,\tau}^{\intercal}X_{0:\tau})\big],$$

where ϕ is the activation function, $w_{k,\tau}$ the weights of the linear (output) layer, and $\mathbf{v}_{k,\tau}$ the weights of the hidden layer. Since X_0 is a constant (it describes the initial conditions of the simulation) there is no need for an explicit bias term since the first component of $\mathbf{v}$ will be the constant term.

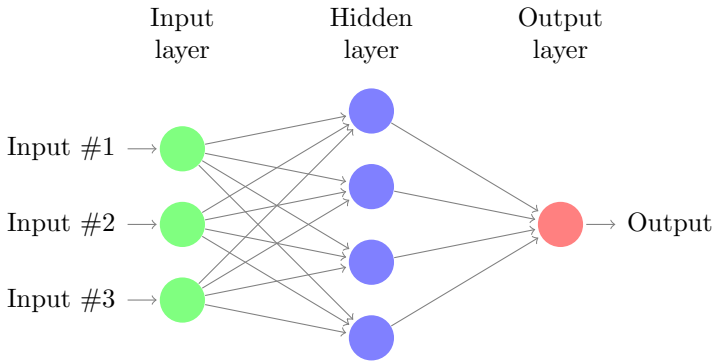


Figure 2.1: *Single-layer perceptron structure*

In this first neural network model, we do not know the distribution of X, which contains arbitrary economic variables, and therefore the risk-neutral expectation cannot be calculated in closed form. When this expectation is required, as in Section 2.6.6, we present results for this model based on Monte Carlo valuation. Since this model's input is X—the vector of economic variables—we will refer to this model as the *nn econ* when showing results.

The second neural network model is more interesting because it allows a closed form valuation. When discussing Equation (2.1), we pointed out that X is modelled as a function of a smaller set of normal random drivers ξ—that is, $X = X(\xi)$ and $\dim(\xi) < \dim(X)$. Using this fact, we express our second model as

$$\widehat{V}_t = E_t^{\mathbb{Q}}\big[\sum_{\tau>t}\sum_{k} w_{k,\tau}\phi(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau})\big];$$

that is, we use ξ as input instead of X (for symmetry, we keep a ξ_o component, which is not random but constant in order to provide a bias term).

Using ξ as input brings two advantages and one disadvantage. On the positive side, the dimensionality of ξ is smaller than that of X ($dim(\xi) < dim(X)$) and its components are uncorrelated with each other ($\sigma(\xi_i, \xi_j) = \delta_{ij}$ but $\sigma(X_i, X_j) \neq \delta_{ij}$). This makes solving the non-linear optimization problem much easier, requiring fewer samples and shorter training

time to converge. Most importantly, the normal distribution of ξ allows a closed-form solution to the risk-neutral expectation as we show later in Theorem 1. On the negative side, $CF'(\xi) = CF \circ X(\xi)$ is a more complex function than $CF(X)$ so—all else being equal—we would expect to need a bigger (larger k) neural network, more samples, and a longer training time. Given these advantages and disadvantages, it is not possible to tell analytically which model will show higher accuracy. Both will, therefore, be tested. Since this model's input is ξ—the vector of normal random variables—we will refer to this model as the *nn rand* when presenting results.

2.4.0.1 The risk-neutral value of a neural network We have claimed that the second neural network model has a closed-form solution to the risk-neutral expectation. We now show its derivation, which, as far as we know, has not appeared before in the literature.

THEOREM 1. *For a normally distributed ξ, the time-t risk-neutral expectation,*

$$E_t^{\mathbb{Q}}\big[w_{0,\tau} + \sum_{k=1}^{K} w_{k,\tau}\phi(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau})\big],$$

of a single-layer network with K hidden nodes and a ReLu activation function ϕ that models the cash-flows at time τ is given by

$$w_{0,\tau} + \sum_{k=1}^{K} w_{k,\tau}\frac{1}{2}\left[{}_t\mu_{k,\tau} + {}_t\sigma_{k,\tau}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{{}_t\mu_{k,\tau}^2}{2{}_t\sigma_{k,\tau}^2}\right) + {}_t\mu_{k,\tau}\left(1 - 2\Phi\left(-\frac{{}_t\mu_{k,\tau}}{{}_t\sigma_{k,\tau}}\right)\right)\right], \quad (2.3)$$

$$\begin{aligned} {}_t\mu_{k,\tau} &= \sum_{i=0}^{\min(t,\tau)} \mathbf{v}_{i,k,\tau}^{\intercal}\xi_i, \\ {}_t\sigma_{k,\tau}^2 &= \sum_{i=t+1}^{\tau} \|\mathbf{v}_{i,k,\tau}^{\intercal}\|^2. \end{aligned}$$

Proof. Starting from the full network

$$E_t^{\mathbb{Q}}\big[w_{0,\tau} + \sum_{k=1}^{K} w_{k,\tau}\phi(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau})\big] = w_{0,\tau} + \sum_{k=1}^{K} w_{k,\tau}E_t^{\mathbb{Q}}\big[\phi(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau})\big], \quad (2.4)$$

and then focusing on the expectation of a single node, we obtain

$$\begin{aligned} E_t^{\mathbb{Q}}\big[\phi(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau})\big] &= E_t^{\mathbb{Q}}\big[\max(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}, 0)\big] \\ &= E_t^{\mathbb{Q}}\big[\frac{\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau} + |\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}|}{2}\big] \\ &= \frac{1}{2}\Big[E_t^{\mathbb{Q}}\big[\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}\big] + E_t^{\mathbb{Q}}\big[|\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}|\big]\Big]. \end{aligned} \tag{2.5}$$

Since $\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}$ is normally distributed, it is defined by its mean and standard deviation, $\mu_{k,\tau}$ and $\sigma_{k,\tau}$. Its conditional expectation at time t is

$$E_t^{\mathbb{Q}}\big[\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}\big] = {}_t\mu_{k,\tau} = \sum_{i=0}^{\min(t,\tau)} \mathbf{v}_{i,k,\tau}^{\intercal}\xi_i,$$

and its conditional variance at time t is

$${}_t\sigma_{k,\tau}^2 = \sum_{i=t+1}^{\tau} \|\mathbf{v}_{i,k,\tau}^{\intercal}\|^2.$$

If $\tau \leq t$, then its conditional variance is 0.

Since $\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}$ is normally distributed, $|\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}|$ follows a folded normal distribution, with conditional expectation at time t

$$E_t^{\mathbb{Q}}\big[|\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau}|\big] = {}_t\sigma_{k,\tau}\sqrt{\frac{2}{\pi}}\exp(-\frac{{}_t\mu_{k,\tau}^2}{2{}_t\sigma_{k,\tau}^2}) + {}_t\mu_{k,\tau}\big(1 - 2\Phi(-\frac{{}_t\mu_{k,\tau}}{{}_t\sigma_{k,\tau}})\big).$$

Therefore, the expectation of each hidden node is

$$E_t^{\mathbb{Q}}\big[\phi(\mathbf{v}_{k,\tau}^{\intercal}\xi_{0:\tau})\big] = \frac{1}{2}\left[{}_t\mu_{k,\tau} + {}_t\sigma_{k,\tau}\sqrt{\frac{2}{\pi}}\exp\left(-\frac{{}_t\mu_{k,\tau}^2}{2{}_t\sigma_{k,\tau}^2}\right) + {}_t\mu_{k,\tau}\left(1 - 2\Phi\left(-\frac{{}_t\mu_{k,\tau}}{{}_t\sigma_{k,\tau}}\right)\right)\right],$$

which leads to the expectation of the full network in (2.3).

□

2.5 Qualitative comparison

Besides the quantitative experiments, whose results we show in Section 2.6.6, there are some qualitative differences of importance to practitioners. One of them is the complexity of the model, as measured by the number of modules and equations required to implement it. A second one is the amount of expert judgement required in the feature engineering. A third one is how to prevent overfitting of the training data.

2.5.1 Model complexity

Figure 2.2 presents a comparison of the module structure. We describe below the sequence of calculations required for the training phase, the prediction phase being very similar with the exception that cash flows are replaced by prices (closed-form or Monte Carlo, depending on the model and the instruments used).

All models require a random number generator. Replicating portfolios and the first neural network model require the generation of the economic variables X. In the insurance industry, these two modules are usually grouped into the so-called economic scenario generator (ESG). The second neural network model does not need X for training or prediction. Finally, the replicating portfolio model requires the generation of instrument cash flows in order to produce the inputs to the optimization problem. Despite their reputation for complexity, neural networks actually lead to a simpler model, at least when measured by the number of equations and components necessary for their implementation.

2.5.2 Feature engineering

Any replicating portfolio model requires a substantial degree of expert judgement in deciding which instrument cash flows to use in the model. When working with simple liabilities, the decision is not hard since the simplest instruments, such as bonds and equity forwards, will work well. As the liabilities grow in complexity, and if the most obvious derivatives (swaps, swaptions, European and Asian options) are not enough to capture the behaviour of the liabilities, the practitioner faces the extremely difficult task of figuring out which is the correct derivative to add to the existing mix. Since financial instruments

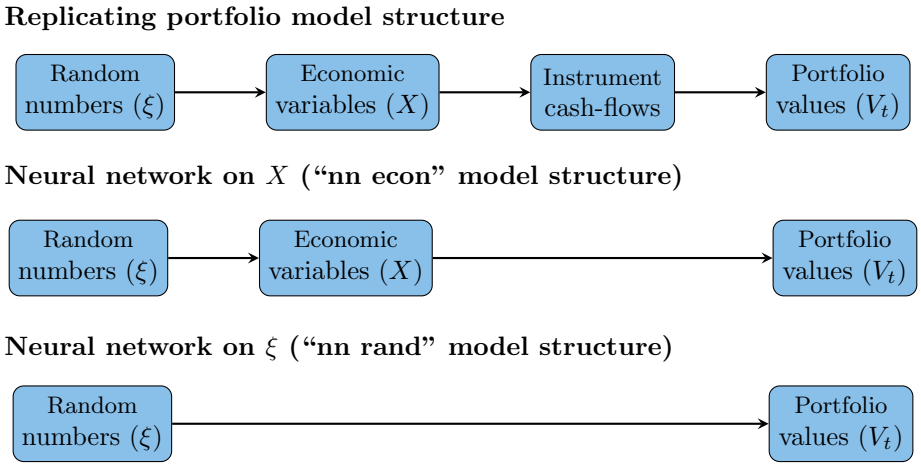


Figure 2.2: *Comparison of model structures*

as a whole do not form a structured basis of any meaningful space of functions, it is not possible to explore in a systematic way the set of all possible financial instruments until the best solution is found. Even worse, each attempt (adding of a new asset class) requires a substantial amount of implementation work before the results can be seen. In contrast, neural networks provide certain guarantees of convergence by simply increasing the width of the hidden layer (adding more nodes). This guarantee is provided by the *universal representation theorem*, of which different versions exist (among them Hornik (1991) and Hanin and Sellke (2017)). At least for the classes of functions covered under these theorems, the search for better results is extremely straightforward: just one parameter that is a direct input in any neural network software library. No new equations need to be implemented, only one input change in the existing model in required. Even when no new asset classes are required, the feature engineering problem in replicating portfolios still exists. Each asset class can have tens, hundreds, or thousands of individual instruments. For example, there is one zero coupon for each possible maturity. Even worse is the case of swaptions: having to choose from a combination of a set of maturities, tenors, and strikes leads to having to select thousands of instruments. All these basis elements must be completely calculated before being fed to the linear regression problem. In contrast, neural networks create their own features adaptively based on the data. This, of course, has the downside that it requires more training data than a comparable replicating

portfolio model for which the expert judgement selection has been carried out correctly.

2.5.3 Feature selection

The reverse problem to not having enough financial instruments or not having asset classes that are complex enough is the problem of having too many instruments and asset classes. Feeding thousands of instruments to the regression problem runs the very real risk of overfitting the training data. This problem has not been entirely solved by practitioners in the industry, although the popularization of machine learning software libraries has allowed big improvements in recent years compared with the completely manual approach used five or ten years ago. In this paper, we solve this problem by using Lasso regression (Tibshirani (1996)) with the Akaike information criterion AIC (Akaike (1998), Burnham and Anderson (2002)) to choose the regularization parameter. While there are many methods for the selection of the regularization parameter (see the introduction to Ninomiya et al. (2016)), we chose the AIC approach with the adjusment described in Zou et al. (2007) as implemented in a popular machine-learning Python library. For the neural network model, the constraints on model complexity are provided by selecting the layer width via cross-validation.

2.6 Numerical experiments

The goal of this section is to provide a quantitative comparison between a neural network model and a replicating portfolio model. It is important to clarify that there are many possible neural network models (many hyper-parameters and architectural choices) as well as many possible replicating portfolio models (many options of asset classes in the instrument universe and other architectural choices). Furthermore, the results presented here are for one particular scenario generator and one insurance product. It is not possible to generalize the conclusions drawn from these results to all models and all products. However, the choices made in this example are mainstream and robust. We would, therefore, expect the conclusions to extend to many real-world situations.

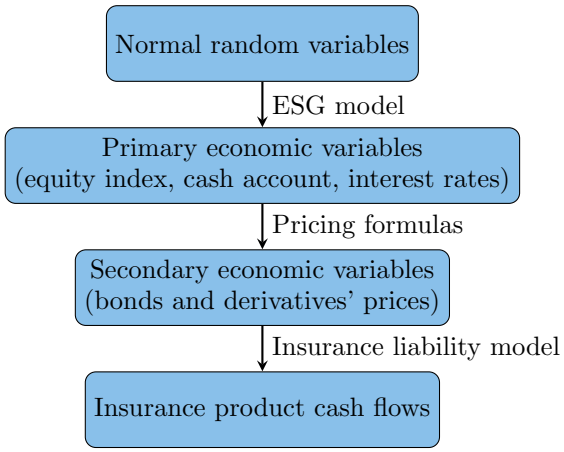


Figure 2.3: *Modular structure of the insurance model*

2.6.1 Experimental setup

In order to provide a comparison between replicating portfolios and another model, it is necessary to first have access to simulated cash flows of an insurance product. In the absence of open source libraries or data sets there has been no option but to build our own economic scenario generator and insurance model. Hoping that the data might help others in their research in the field, we have published the full data set in Mendeley Data (Fernandez-Arjona (2019)).

The high-level structure of the insurance model is described in Figure 2.3. The first module is the normal random variable generator, which feeds the second module, the economic scenario generator. We use a combination of a one-factor Hull–White for the short rate and a geometric Brownian motion process for equity returns (in excess of risk-free rates). For the interest rate model, we use the formulas in Glasserman (2013).

In addition to the published data set, the full code of the scenario generator is available in open source form at `https://gitlab.com/luk-f-a/EsgLiL`. The generator is entirely written in Python. It uses NumPy (Oliphant (06), Van Der Walt et al. (2011)) for array operations, pandas (McKinney (2010)) for data aggregation, scikit-learn (Pedregosa et al. (2011a)) for linear regressions and neural networks training, and joblib for parallelization. This scenario generator produces the evolution of the short rate, the cash account and the equity index. From these, we derive the rest of the asset prices: bonds, swaptions, and

equity options. The last module is the insurance product, a variable annuity guarantee known as *guarantee return on death*. The simulation of the insurance product begins with a policyholder population of 1,000 customers of ages 30 to 70. The population evolves according to a Lee–Carter stochastic mortality model (we have used the same parameters as in Lee and Carter (1992)), which provides a trend and stochastic fluctuations. Each customer starts the simulation with an existing investment fund and a guaranteed level. In each time period, they pay a premium, which is used to buy assets; these assets are deposited in the fund. The fund is rebalanced at each time period to maintain a target asset allocation. The value of the fund is driven by the inflows from premiums and the market value changes are driven by the interest rate, equity, and real estate models. At each time step, a proportion of policy holders (as determined by the stochastic life table) die 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 increase the guaranteed amount for each policy.

All policies have the same maturity date, of forty years. Those policyholders alive at maturity receive the investment fund or the guaranteed value, whichever is the higher.

2.6.2 Ground truth and benchmark value

The quality comparison across methods requires establishing a *ground truth*—the values of the risk metrics calculated in an exact way. Given the lack of closed-form formulas this is not possible, so we settle for performing comparisons against a benchmark value calculated in the most reliable way possible. We do this by running an extremely large Monte Carlo simulation, with 100,000 outer simulations each with 10,000 inner simulations.

The results of this calculation are shown in Table 2.1. The centre column shows the mean present value, the risk-neutral value of the guarantee. To the left and right we see the expected shortfall and value at risk, at a confidence level of 1% and 99% for *left* and *right* respectively. Both expected shortfall and value at risk are expressed as the change

in monetary value in respect of the mean—that is as $V_1^{tail} - V_0$.

Table 2.1: *Benchmark values*

	Left ES	Left VaR	Mean	Right VaR	Right ES
Large nMC	-3.9×10^6	-3.3×10^6	-5.5×10^6	2.5×10^6	2.7×10^6

2.6.3 Nested Monte Carlo and replicating portfolio benchmarks

We use two established methods to benchmark our proposed model: nested Monte Carlo (nMC) and replicating portfolios (RPs). In all cases we set a computational budget of 10,000 training samples. In the case of nested Monte Carlo, this budget is split into 100 outer and 100 inner simulations. This split was selected for being the combination with the largest number of inner simulations possible (lowest bias) with a minimum of 100 outer simulations (to be able to calculate the 1 percent expected shortfall). For reference, note the large difference with the benchmark value calculation made with $100{,}000 \times 10{,}000$ simulations. Since the training samples are randomly drawn, each estimator is itself a random variable and we treat them as such. For each method, we calculate 100 macro-runs with new random numbers in order to obtain an empirical distribution of the estimators.

For the given sample budget of 10,000 simulations, the nested Monte Carlo estimator requires choosing the mix between outer and inner simulations. A higher number of outer simulations decreases variance, but a lower number of inner simulations increases bias. This effect is known as the bias–variance trade-off. Figure 2.4 shows this effect quite clearly using a violin plot, showing the increasing bias and decreasing variance from left to right. Each individual distribution is annotated with a percentage showing the mean absolute percentage error of the estimator. We describe this error measure in more detail in the next section.

Regarding the replicating portfolio benchmark, it is important to note that there is not *one* replicating portfolio method that we could directly apply. Replicating portfolios is a method with many variations and requires the choice of a range of hyper-parameters.

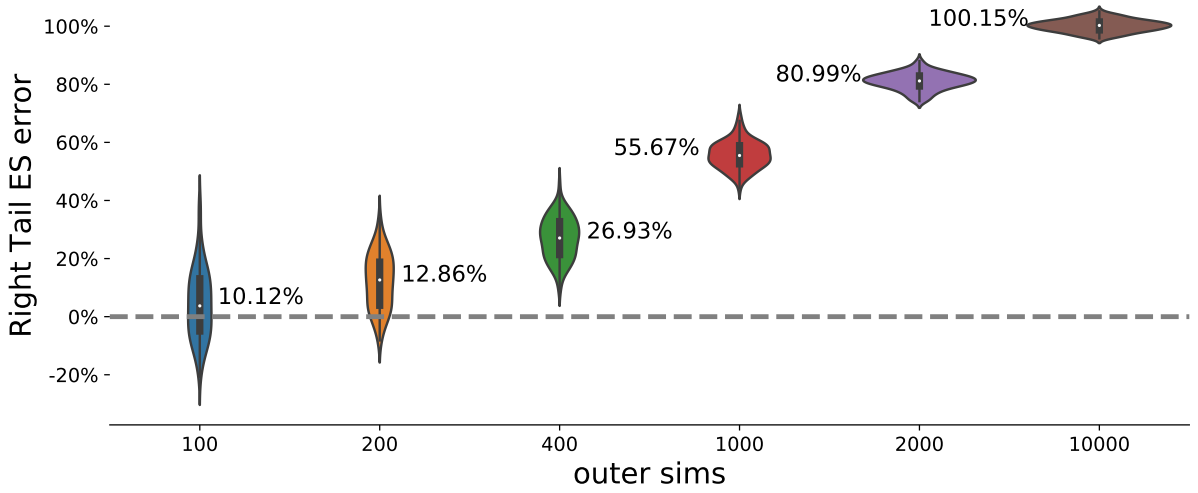


Figure 2.4: *(colour online) Error distribution of nMC estimators (fixed total simulations). Increasing outer simulation numbers lead to reduced variance but the corresponding decrease in inner simulation numbers increases bias.*

Table 2.2: *Replicating portfolio parameters*

Parameter	**Choice**
Loss function	Squared errors
Asset universe	Bonds, cash, swaptions, equity index, equity European options
Time steps	Full annual cash flow replicating (no grouping)
Constraints	None
Optimizer	LassoLarsIC (scikit-learn)

Which explains the importance both of the description that follows and of our choices with regard to implementation. We have chosen parameters common in the industry except for one aspect. In the industry, replicating portfolio models are usually run many times with different parameters (in particular different asset classes) and one replicating portfolio is selected manually from those runs. Following that methodology would not allow us to create repeatable experiments or to form consistent distributions. We therefore use a Lasso regressor for the feature selection (instrument selection from the universe) and use the Akaike information criterion to choose the regularization parameter. This is provided by the machine learning library scikit-learn through the LassoLarsIC class. As for the rest of the parameters, we list them in Table 2.2.

Table 2.3: *Neural network model parameters*

Parameter	Choice
Loss function	Squared errors
Architecture	Single-layer perceptron, 100 nodes, ReLu activation function
Time steps	Full annual cash flow replicating (no grouping)
Constraints	None
Optimizer	L-BFGS (scikit-learn)

2.6.4 Neural network model tuning and calibration

Since the number of layers and activation function are given, there is only one hyperparameter in the neural network model, k. Traditionally, this parameter would be tuned using cross-validation. In our case, the model showed such good results with a low number—$k = 100$—that we have only performed a basic sensitivity analysis with $k = 50$ and $k = 200$, which showed that results did not drastically improve in either case. No further tuning was necessary.

The calibration of the neural network model was performed using scikit-learn's L-BFGS implementation.

Table 2.3 summarizes the most important neural network parameters.

2.6.5 Quality measurement

Following the arguments in Willmott and Matsuura (2005) and Chai and Draxler (2014) we choose to focus on absolute errors rather than squared errors for model comparison since we do not need to penalize large outliers, and the errors are biased and most likely do not follow a normal distribution. We therefore use mean absolute errors as a metric of mean model errors. Other moments are not quantitatively measured but the histograms of the distributions are presented for qualitative assessment.

The error to be considered is that of each risk metric (ES or VaR) and each tail (left tail or right tail), measured as a percentage error against the benchmark value for that metric.

Based on the 100 macro-runs described in the previous section, we derive an empirical

distribution for each estimator, $\{\rho_i\}_{i=1}^{100}$ where each individual sample is denoted as ρ_i. The mean absolute percentage error (MApE) is defined as

$$\frac{1}{100}\sum_{i}^{R}\left|\frac{\widehat{\rho_i}}{\rho}-1\right|,$$

where ρ is the benchmark value of the estimator.

All model results are mean-centred before risk metrics are calculated.

Figure 2.4 shows an example of the application of this error measurement for the various parameters in the nested Monte Carlo estimator. We can see that the estimator with 100 outer and 100 inner simulations has the lowest MApE. We therefore use this estimator in all subsequent comparison with other methods.

It is important to note that the benchmark value calculation is completely out-of-sample in respect of the training of any of the methods. Hence, all comparisons shown below are fully out-of-sample.

2.6.6 Results

The results of the numerical experiments are presented in Table 2.4 and Figure 2.6. The columns show the mean absolute percentage error for each of the four risk metrics and the mean present value. The rows show the errors of each different method.

The errors of the replicating portfolio model are shown in the first row. In terms of mean absolute errors, this method yields the worst results of the group. Interestingly, the results are worse than using a nested Monte Carlo approach (second row). This is probably due to the very complex and non-linear nature of the guarantee function that is being replicated. Most likely the asset classes in the instrument universe are not complex enough, or not sufficiently path dependent to replicate the guarantee.

While one might consider adding more asset classes, it becomes immediately clear that there is no obvious next step. This is a key disadvantage of replicating portfolios versus polynomial or neural network methods: the complexity of the approximation (richness of the approximating function) is not a parameter that one can easily modify. Which asset

Table 2.4: *Comparison of errors measured as MApEs*

	Left ES	Left VaR	Mean	Right VaR	Right ES
Rep. Portfolio MApE	20%	23%	4%	46%	47%
Nested MC MApE	19%	14%	2%	8%	10%
Neural net (econ) MApE	4%	2%	2%	7%	4%
Neural net (rand) MApE	15%	11%	5%	4%	5%

class to add next is an unknown, and there are hundreds of exotic derivatives that one might try. In order to test any of them, one must program the cash flows and valuation functions before any testing can be done. Therefore, the search for a better model takes much longer and success is not even guaranteed since one might not find the correct derivative.

By contrast, when working with a polynomial approximation one might increase the maximum degree of the polynomial, and with a neural network one might add layers or make each layer wider (adding more nodes). It only takes a few keystrokes to make a more powerful model. Polynomial and neural network models are guaranteed to succeed under certain conditions (smooth enough functions, sufficient samples, etc.), at least asymptotically. These models are, therefore, more amenable to automated solutions than are replicating portfolio models, which require an expert setup.

The last two rows in Table 2.4 present the results for each type of neural network models. Each model shows better results than nested Monte Carlo or replicating portfolios. The first type (economic variable inputs) shows the best results of the group, better than the second type (random variable inputs).

Despite the advantages of the second type of neural network model, the data shows that 10,000 samples were not enough to learn the more complex function $CF \circ X$ sufficiently well to have higher accuracy than the first type of neural network model, which only had to learn the simpler function CF. In Figure 2.5 we can see that the second model (called *NNrand:10* when trained on 10,000 samples, *NNrand:50* when trained on 50,000 samples, etc.) does perform better than the first model (*NNecon*) once it is given a larger number of samples.

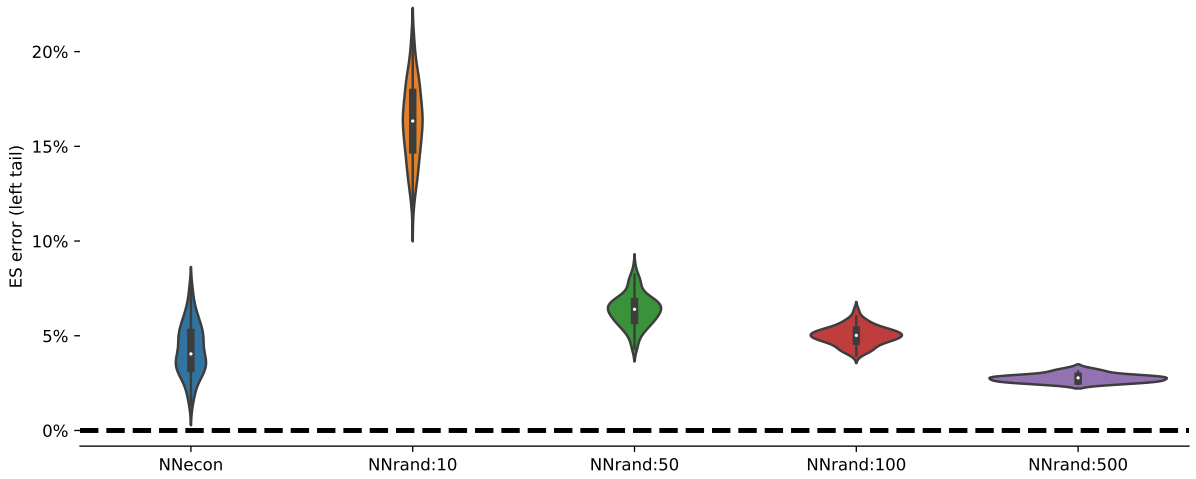


Figure 2.5: *(colour online) Effect of increasing training sample size on second neural network model. The decrease in mean and standard deviation of errors can be clearly seen as samples are increased from 10,000 to 500,000.*

Figure 2.6 shows the empirical distribution of each of the estimators (trained, respectively, on 10,000 samples for comparability). We can observe the relative standard deviations of the estimators and find, as previously seen in Figure 2.4, that nested Monte Carlo has low bias and a very high variance compared to the other methods. Interestingly, the standard deviations of the neural network models are not much higher than that of the replicating portfolios, despite each macro-run using a completely different set of inputs. This indicates that the calibration of the neural networks is robust to the variance of the inputs. In this regard, neural networks have a bad reputation due to their non-convex loss function. A common problem associated with the training of neural networks is that of local minima. Together with the common use of stochastic optimization methods (such as stochastic gradient descent), this usually contributes to a high variance of predictions. In this paper, we have taken certain steps to reduce this problem, by a) using a network of a small size (100 nodes) and b) using the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimization algorithm, which is non-stochastic. These decisions contribute to keeping the variance of the estimator within reasonable levels.

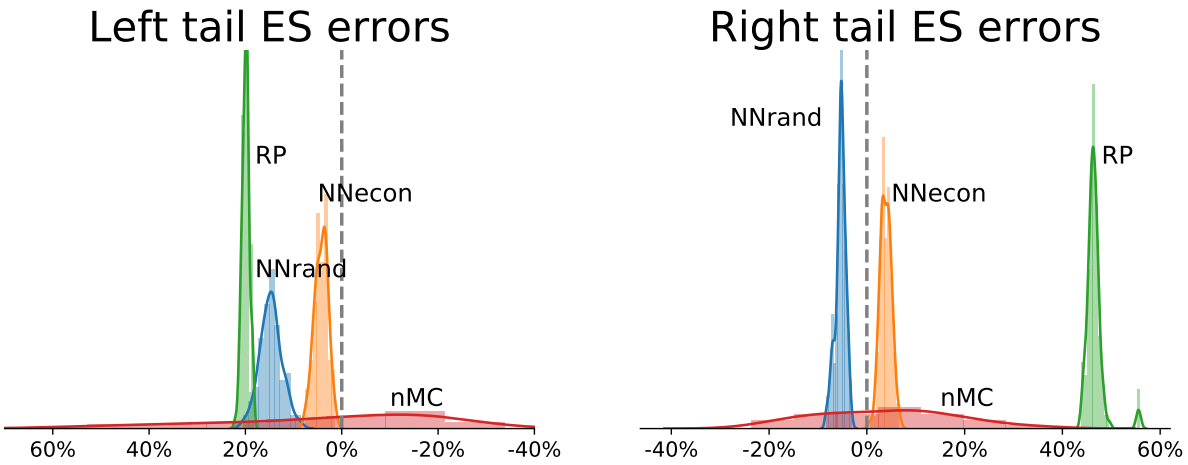


Figure 2.6: *(colour online) Empirical distribution of estimators (with 10,000 training samples). Each neural network model delivers more accurate results than the benchmark models.*

2.6.7 Runtime

As shown in Table 2.4 and Figure 2.6, the neural network models perform very well in terms of quality. Since these models require a non-linear regression to find their parameters, they can be expected to be slower than a purely linear model, such as a replicating portfolio. Table 2.5 shows how long it takes to run each model *end to end*—that is, including feature generation, calibration and prediction.

Table 2.5: *Comparison of runtime (in seconds) for training data sets of different sizes*

Samples	Neural net (econ)	Neural net (rand)	Rep. portfolio
2500	112	115	4
5000	202	200	4
10000	384	406	6
20000	707	711	13
40000	1458	1407	26

We can see that training a neural network model takes longer than training a replicating portfolio. Both types of model scale approximately linearly to the size of the training set. However, even at the far end (40,000 training samples) the neural network models do not take more than 30 minutes (on a single-core AMD Opteron 6380) which is perfectly acceptable from a practitioner's point of view. Generating the inputs required for economic capital calculations can normally take from several hours to several days; and 30 minutes can therefore fit easily within the normal production schedule of an insurance

company.

2.7 Conclusions

Based on a simulated insurance product, we have presented a comparison of nested Monte Carlo, replicating portfolios, and neural networks as methods for calculating solvency capital. The numerical experiments show that neural networks perform very well, even in a highly non-linear problem with a small number of training samples. The mean errors are the lowest of the group and the distributions of the results do not show qualitatively, large variance. A qualitative analysis suggests that the neural network model can also have advantages in terms of model simplicity.

In the construction of this neural network model we make use of what we believe is a novel formula for calculating the risk-neutral price of a neural network. Additionally, we make two contributions for other researchers in the field: a description of an automated replicating portfolio model (necessary for reliable comparisons) and a full data set and software library for the production of economic scenarios.

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

Lucio Fernandez-Arjona, University of Zurich

Damir Filipović, EPFL and Swiss Finance Institute

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.

3.1 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 use-

We thank participants at the SIAM Activity Group on Financial Mathematics and Engineering virtual seminar, Ben Feng, Antoon Pelsser, two anonymous referees, and the associate editor for their comments.

ful 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 dimension, 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 (Glasserman and Yu (2002)) given the regression is made against terminal payoffs and not against empirical conditional expectations.

We implement two numerical examples of high-dimensional Gaussian models—a plain 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, Fernandez-Arjona and Filipović (2020). We hope that these datasets can be used by others to allow more direct comparisons between methods in future research.

3.1.1 Related literature

An early example of static replication via basis functions can be found in Madan and Milne (1994), which presents a framework for replication of general contingent claims. These contingent claims are modeled in a Hilbert space and the static replication problem is solved by constructing a countable orthonormal basis. The method is applied to the pricing and hedging of these contingent claims. Carriere (1996) and Longstaff and Schwartz (2001) use a sequential approximation algorithm to calculate the conditional expectations required in the valuation of options with 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. Andreatta and Corradin (2003) apply this idea to valuation of life insurance policies.

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

While Glasserman and Yu (2002) 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 Pelsser and Schweizer (2016)—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, Broadie et al. (2015) 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 Pelsser and Schweizer (2016). Path-dependency and the resulting high-dimensional problem in long-term projections is managed via a hand-picked dimensionality-reduction function to avoid the curse of dimensionality. In that framework, the basis functions are not guaranteed to have a closed form solution, whose existence depends on the choice of dimensionality-reduction 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 Duong (2019). In this case, the method is based on splines in a regress-now setting. In contrast, our models are based on global polynomials in a regress-later setting.

A neural network model is applied to solvency capital problem in life insurance in Castellani et al. (2018). The neural network model shows better performance than LSMC—regress-now model with polynomial basis functions—and a support vector regression model. All three models—including the neural network model—are regress-now mod-

els. By contrast, we focus on regress-later methods, which show better accuracy in the examples in addition to being better in theory.

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

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

Table 3.1 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.

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, Hokanson and Constantine (2018) showed that polynomial ridge approximation can be extended to reduce dimensionality in a data-driven manner. We follow a similar approach and apply it to portfolio pricing and risk management.

Table 3.1: *Comparison of regress-now and regress-later methods in the literature*

	Dimensionality reduction on inputs	**Regression method**
Regress-now		
Broadie et al. (2015)	none	incomplete, manually-selected monomial basis
Duong (2019)	none	polynomial spline space
Castellani et al. (2018)	implicit in neural network	neural network
Ha and Bauer (2020)	none	left singular functions of the conditional expectation operator
Hong et al. (2017)	model decomposition	kernel smoothing
Risk and Ludkovski (2018)	none	Gaussian process regression
Regress-later		
Pelsser and Schweizer (2016)	manually selected function	orthonormal piecewise linear basis and Hermite polynomials
this paper	linear dimensionality reduction	orthogonal polynomials and shallow neural networks

Nested Monte Carlo methods for portfolio management have been studied in Lee and Glynn (2003) and Gordy and Juneja (2010) among others. Gordy and Juneja (2010) show several methods that can reduce the computational cost of 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 LeCun et al. (1998). 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 3.2 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 3.3 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 3.4 and 3.5 provide numerical case studies: the initial European call option example revisited in Section 3.4, and an insurance liability model in Section 3.5. Section 3.6 concludes. The appendix contains proofs and technical background material. Section 3.7 describes the quality metrics used to compare different methods. Section 3.8 contains the economic scenario generator underlying the numerical examples in the main text. Section 3.9 contains all proofs and some auxiliary results of independent interest. Sections 3.10 and 3.11 present an analysis of the sensitivity of the proposed methods to different hyper-parameters.

3.2 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 $X = (X_1, \dots, X_T)$ 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 $\mathcal{F}_t = \sigma(X_1, \dots, X_t)$, $t = 1, \dots, T$, 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 $\zeta_t = \zeta_t(X_1, \dots, X_t)$. 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, insurance liabilities, mortgage-backed instruments and other structured products. In most real-world cases, V_t is not given in closed form but has to be estimated from simulating $f(X)$. This creates computational challenges, as the function f may be costly to query and available computational budget is limited.

There are several risk management applications of the portfolio value process. We focus here on risk measurement. Insurance and banking solvency regulatory frameworks, such as Solvency II, Swiss Solvency Test, and Basel III, require capital calculations that are based on risk measurements of the value changes $\Delta V_t = V_t - V_{t-1}$ over risk periods $(t-1, t]$. The most common risk measures are *value at risk*

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

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., (Föllmer and Schied, 2004, 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 3.1. 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.

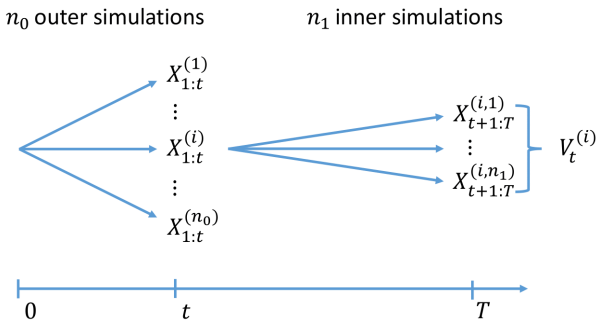


Figure 3.1: *Nested Monte Carlo structure. The outer simulations span the time period 0 to t, and each of those serves as the starting point of a group of inner simulations.*

3.2.1 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 3.8. That generator maps X to a vector of economic factors. This vector contains several components, among them the equity index EQ_t 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—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

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

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 3.2 and 3.3 for a fixed total budget of 50,000 simulations. In each individual estimation (inner–outer combination), the error is calculated as a percentage of the benchmark value and therefore we refer to the quality metric as *MApE*, for Mean Absolute percentage Error. The formula is described in Section 3.7. 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 $\mathrm{ES}_{99\%}(-\Delta V_1)$.

Table 3.2: *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 3.3: *Nested Monte Carlo, comparison of expected shortfall for nested Monte Carlo (50,000 total samples)*

	Benchmark	Nested Monte Carlo (MApE by inner simulations)							
Maturity	(value)	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 Broadie et al. (2015), 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

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.

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 3.4. There, for example, the error on the 5 year call is 1.7% while in Table 3.2 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 3.4: *Nested Monte Carlo, comparison of present value and expected shortfall MApE (in percentage points) for optimal inner–outer split*

	Present Value		**Expected Shortfall**	
Samples	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 3.4 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.

3.3 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 $L^2_{\mathbb{Q}}$ that is spanned by an optimally chosen set of basis functions in X, which constitute the feature map. We assume that these basis functions admit conditional expectations in closed form. We thus obtain, as second step, an approximation of the portfolio value process in closed form. In practical applications, we learn the approximation of f from a finite sample of X, which induces an empirical measure that proxies the model population measure $\mathbb{Q}$. The performance of this approach hinges on the choice of the basis functions. We formalize and discuss all this in detail in the following.

3.3.1 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 $L^2_{\mathbb{Q}}$, for $i = 1, \dots, m$. These functions form the *feature map* $\phi_\theta = (\phi_{\theta,1}, \dots, \phi_{\theta,m})^\top$. For any $\theta \in \Theta$, the $L^2_{\mathbb{Q}}$-projection of f on $\operatorname{span}\{\phi_{\theta,1}, \dots, \phi_{\theta,m}\}$ 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

$$\min_{\beta \in \mathbb{R}^m} \|f - \phi_\theta^\top \beta\|_{L^2_{\mathbb{Q}}}. \tag{3.1}$$

We assume that the conditional expectations $\mathbb{E}^{\mathbb{Q}}_t[\phi_\theta(X)] = G_{\theta,t}(X_1, \dots, X_t)$ 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

$$G_{\theta,t}(x_1, \dots, x_t) = \mathbb{E}^{\mathbb{Q}}[\phi_\theta(x_1, \dots, x_t, X_{t+1}, \dots, X_T)], \tag{3.2}$$

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

$$V_{\theta,t} = G_{\theta,t}(X_1, \dots, X_t)^\top \beta_\theta, \tag{3.3}$$

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 Glasserman and Yu (2002). 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 (3.3)? By Doob's inequality, making use of the martingale property of the value process, we obtain an upper bound on the pathwise maximal $L^1_{\mathbb{P}}$-error,

$$\|\max_{t\leq T}|V_t - V_{\theta,t}|\|_{L^1_{\mathbb{P}}} \leq \|\tfrac{d\mathbb{P}}{d\mathbb{Q}}\|_{L^2_{\mathbb{Q}}}\, \|\max_{t\leq T}|V_t - V_{\theta,t}|\|_{L^2_{\mathbb{Q}}} \leq 2\|\tfrac{d\mathbb{P}}{d\mathbb{Q}}\|_{L^2_{\mathbb{Q}}}\|f - f_\theta\|_{L^2_{\mathbb{Q}}}. \tag{3.4}$$

Note that $\|\frac{d\mathbb{P}}{d\mathbb{Q}}\|_{L^2_{\mathbb{Q}}}$ is known by the modeler. The $L^2_{\mathbb{Q}}$-approximation error on the right hand side of (3.4) is the objective in (3.1). 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 $L^1_{\mathbb{P}}$-error of the approximation of V_t.

3.3.2 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

$$\min_{(\theta,\beta)\in\Theta\times\mathbb{R}^m} \|f - \phi_\theta^\top \beta\|_{L^2_{\mathbb{Q}}}. \tag{3.5}$$

Here is an elementary existence result.

LEMMA 1. *Assume*

(i) Θ *is compact,*

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

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

Then there exists a solution to (3.5).

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* $f \in L^2_{\mathbb{Q}}$ *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 (3.5) 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

$$\phi_{\theta,i}(x) = g_i(A^\top x + b) \tag{3.6}$$

for some exogenously given functions $g_i : \mathbb{R}^p \to \mathbb{R}$, $i = 1, \dots, m$, 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 $X = (X_1, \dots, X_T)$, we decompose the $dT \times p$-matrix A into T consecutive blocks $A_t \in \mathbb{R}^{d\times p}$ such that $A^\top = (A_1^\top, \dots, A_T^\top)$. Then we have $A^\top \operatorname{vec}(X) = \sum_{t=1}^{T} A_t^\top X_t$ and

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

The conditional expectation functions in (3.2) read component-wise for $i = 1, \dots, m$ as

$$G_{\theta,t,i}(x_1, \dots, x_t) = \mathbb{E}^{\mathbb{Q}}[g_i(b + \sum_{s=1}^{t} A_s^\top x_s + \sum_{s=t+1}^{T} A_s^\top X_s)]. \tag{3.7}$$

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 (3.5). In the sequel, we study three different specifications of the type (3.6) in more detail: a full polynomial basis, a polynomial feature map with linear dimensionality reduction, and a shallow neural network.

3.3.3 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 (3.5) boils down to the projection (3.1). We let the feature map ϕ be composed of a basis of the space of all polynomials of degree δ or less,

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

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

$$\mathbb{E}_{\mathbb{Q}}[\|X\|^{2\delta}] < \infty. \tag{3.8}$$

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,

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

Table 3.5 shows that the dimension m quickly becomes larger than the training sample

size in practice.

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

Table 3.5: *Dimension* $m = \dim \mathrm{Pol}_\delta(\mathbb{R}^{dT})$ *for* $\delta = 3$.

The exact form of the conditional expectation (3.2) 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 $L^2_{\mathbb{Q}} = \otimes_{t=1}^T L^2_{\mathbb{Q}_t}$ can greatly simplify the calculations. In view of (Sullivan, 2015, 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

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

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 $\{1, \ldots, m\} \ni i \mapsto \boldsymbol{\alpha}^i \in \{\boldsymbol{\alpha} \in \mathbb{N}_0^{dT} \mid |\boldsymbol{\alpha}| \leq \delta\}$, then we obtain an orthogonal basis of $\mathrm{Pol}_\delta(\mathbb{R}^{dT})$ for $\mathbb{Q}$ by setting $\phi_i = g_{\boldsymbol{\alpha}^i}$. The conditional expectation functions (3.7) are then given in closed form as

$$G_{t,i}(x_1, \ldots, x_t) = \prod_{s=1}^{t} h_{s,\boldsymbol{\alpha}_s^i}(x_s) \prod_{s=t+1}^{T} 1_{\boldsymbol{\alpha}_s^i = \mathbf{0}}. \tag{3.9}$$

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

Strictly speaking, the dimension of the linear span of the functions ϕ_i in $L^2_{\mathbb{Q}}$ could be less than m, because they may be linearly dependent as elements in $L^2_{\mathbb{Q}}$. This is in particular the case when $\mathbb{Q}$ is an empirical measure from a sample of size n, as described in Subsection 3.3.6. In this case, $m > n$ would simply lead to an exact interpolation of f, which likely will result in overfitting.

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.

3.3.4 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 (3.6) for weight matrices A with full rank and bias vectors b. As above, we assume that (3.8) holds, so that $\phi_{(A,b),i} \in L^2_{\mathbb{Q}}$. The following theorem shows that we can assume that $b = 0$ and A lies in the *Stiefel manifold* $V_p(\mathbb{R}^{dT}) = \{A \in \mathbb{R}^{dT \times p} \mid A^\top A = I_p\}$, the set of all orthonormal p-frames in $\mathbb{R}^{dT}$.

THEOREM 2. *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 2 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 3. *For the polynomial feature map, there exists a minimizer in $\Theta = V_p(\mathbb{R}^{dT})$ of (3.5). 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 (3.5) corresponds to a *linear dimensionality reduction* with matrix manifold $V_p(\mathbb{R}^{dT}) \times \mathbb{R}^m$ in the spirit of Cunningham and Ghahramani (2015). The dimensionality reduction is produced exclusively by the linear mapping A of $\mathbb{R}^{dT}$ 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 (3.5) 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 3.6 shows that the total dimension of (3.5) remains

moderate compared to the corresponding figures of the full polynomial basis from Table 3.5.

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

The calculation of the conditional expectation (3.2) is not as simple as for the full polynomial basis in (3.9). Instead we need to compute the unconditional moments (3.7), which here reduce to

$$G_{A,t,i}(x_1, \ldots, x_t) = \mathbb{E}^{\mathbb{Q}}[g_i(\textstyle\sum_{s=1}^t A_s^\top x_s + \sum_{s=t+1}^T A_s^\top X_s)], \tag{3.10}$$

for $A \in V_p(\mathbb{R}^{dT})$. Evaluation of (3.10) 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 (Kan, 2008, Lemma 1), which relates multivariate moments to univariate moments, generalizing $4y_1y_2 = (y_1 + y_2)^2 - (y_1 - y_2)^2$, by

$$y^{\boldsymbol{\alpha}} \equiv y_1^{\alpha_1} \cdots y_p^{\alpha_p} = \tfrac{1}{|\boldsymbol{\alpha}|!} \textstyle\sum_{\boldsymbol{\nu}=\mathbf{0}}^{\boldsymbol{\alpha}} (-1)^{|\boldsymbol{\nu}|} \binom{\alpha_1}{\nu_1} \cdots \binom{\alpha_p}{\nu_p} (h_{\boldsymbol{\alpha},\boldsymbol{\nu}}^\top y)^{|\boldsymbol{\alpha}|}, \tag{3.11}$$

for vectors $h_{\boldsymbol{\alpha},\boldsymbol{\nu}} = (\alpha_1/2 - \nu_1, \ldots, \alpha_p/2 - \nu_p)^\top$. The sum in (3.11) 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 (3.10) 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* $\mathrm{vec}(X) \sim N(0, I_{dT})$ *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 (Kan, 2008, Proposition 2).

3.3.5 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 (3.6) 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 (3.8) holds for $\delta = 1$, so that $\phi_{(a_i,b_i)} \in L^2_{\mathbb{Q}}$. 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 4. *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*

$$\operatorname{span}\{\phi_{(a_1,b_1)}, \ldots, \phi_{(a_m,b_m)}\} = \operatorname{span}\{\phi_{(\tilde{a}_1,\tilde{b}_1)}, \ldots, \phi_{(\tilde{a}_m,\tilde{b}_m)}\} \tag{3.12}$$

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*

$$\{\phi_{(a_1,b_1)}, \ldots, \phi_{(a_m,b_m)}\} \text{ is linearly independent.} \tag{3.13}$$

Note that the converse implication in Theorem 4(i) is not true, as can easily be seen from the case where $m = 1$ and $(\tilde{a}_1, \tilde{b}_1) = -(a_1, b_1) \in S_{dT}$ with $a_1 \neq 0$. What's more, the following example shows that the assumptions in Theorem 4(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,*

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

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 5. *For the shallow ReLU neural network, there exists no minimizer of* (3.5) *in general. Moreover, uniqueness does not hold, in the sense that the optimal subspace* $\operatorname{span}\{\phi_{(a_1,b_1)}, \ldots, \phi_{(a_m,b_m)}\}$ *is not unique, in general.*

The conditional expectation functions (3.7) read here as

$$G_{(a_i,b_i),t}(x_1, \ldots, x_t) = \mathbb{E}^{\mathbb{Q}}[(b_i + \textstyle\sum_{s=1}^{t} a_{i,s}^\top x_s + \sum_{s=t+1}^{T} a_{i,s}^\top X_s)^+], \tag{3.14}$$

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 (3.14) 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* $\operatorname{vec}(X) \sim N(0, I_{dT})$, *we have*

$$Y \sim N(b_i + \textstyle\sum_{s=1}^{t} a_{i,s}^\top x_s, \sum_{s=t+1}^{T} \|a_{i,s}\|^2). \tag{3.15}$$

We then obtain a closed form expression for (3.14) *by combining* (3.15) *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., (Delbaen and Schachermayer, 2006, Section 4.3) or Fernandez-Arjona (2021).*

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

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

Hence the evaluation of (3.14) reduces to the computation of the line integral

$$G_{(a_i,b_i),t}(x_1,\ldots,x_t) = \tfrac{1}{2\pi} \int_{\mathbb{R}} \widehat{F}_Y(w+\mathrm{i}\lambda) \tfrac{1}{(w+\mathrm{i}\lambda)^2}\, d\lambda, \tag{3.16}$$

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 (3.16) are routinely computed in finance applications, e.g, in Lévy type or affine models, Duffie et al. (2003). So one can draw on existing libraries of computer code.

3.3.6 Finite-sample estimation

While surprising and remarkable, the non-uniqueness and non-existence results in Theorems 3 and 5 for polynomial feature maps with dimensionality reduction and shallow ReLU neural networks, respectively, are mainly of theoretical interest, arguably. In practice, we solve (3.5) 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 (3.5) boils down to the projection (3.1), and we obtain the optimal

$$\widehat{\beta} = (\tfrac{1}{n}\Phi^\top\Phi)^{-1}(\tfrac{1}{n}\Phi^\top y) \tag{3.17}$$

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 (3.1) 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., Boudabsa and Filipović (2019).

For the polynomial feature map with dimensionality reduction, we use the Riemannian BFGS algorithm Huang et al. (2015) to find a local minimizer of (3.5) 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 Pedregosa et al. (2011b) for the Python programming language to find a local minimizer of (3.5) 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.

3.4 European call option example revisited

Having presented the theoretical background, we now turn back to our initial illustrating example that we introduced in Section 3.2.1. We apply the functional bases described in Sections 3.3.3, 3.3.4, and 3.3.5, following the steps outlined in Section 3.3.6. For each functional basis we show the same quality metrics as in Section 3.2.1 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 3.7.

3.4.1 Results

The first comparison uses the full polynomial basis described in Section 3.3.3. In Tables 3.7 and 3.8 we present the MApE comparison among nested Monte Carlo (nMC), regress-now polynomial basis and the replicating martingale full polynomial basis. Table 3.7

shows results for the present value and Table 3.8 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 3.9. We can see that Table 3.8 confirms the conclusions from Table 3.9, 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 3.7: *European call, comparison of present value MApE (in percentage points)*

		Full Polynomial basis		**LDR**	**Neural Network**	
Samples	**nMC**	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 3.8: *European call, comparison of expected shortfall MApE (in percentage points)*

		Full Polynomial basis		**LDR**	**Neural Network**	
Samples	**nMC**	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 3.7 and 3.8 do not show results for the full polynomial basis under the replicating

Table 3.9: *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

martingale approach for $T = 40$. The reason for this is the combinatorial explosion in the number of basis functions as the dimensionality of the problem grows. As shown in Table 3.5, 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 3.3.4. Table 3.6 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 $L^2_{\mathbb{Q}}$ can be approximated with arbitrary precision.

The optimization problem in (3.5) 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 2. This reduces the effective dimensionality of the problem and simplifies the calculation of the conditional expectation

of $\phi(X)$ in Equation (3.3). For these examples we have used the Riemannian BFGS algorithm from Huang et al. (2015) using the C++ library published by the authors. Additionally, we tested another two algorithms, Riemannian Trust Regions (Absil et al. (2007)) as implemented by the Python library *pymanopt* (Townsend et al. (2016)), and Grassmann Gauss-Newton (Hokanson and Constantine (2018)) 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 3.10, and a description of the methodology for the sensitivity analysis is found in 3.11. 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 3.10, 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 3.7 and 3.8 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 3.3.5. 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 (Pedregosa et al. (2011b)). The results in Tables 3.7 and 3.8 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 3.2 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.

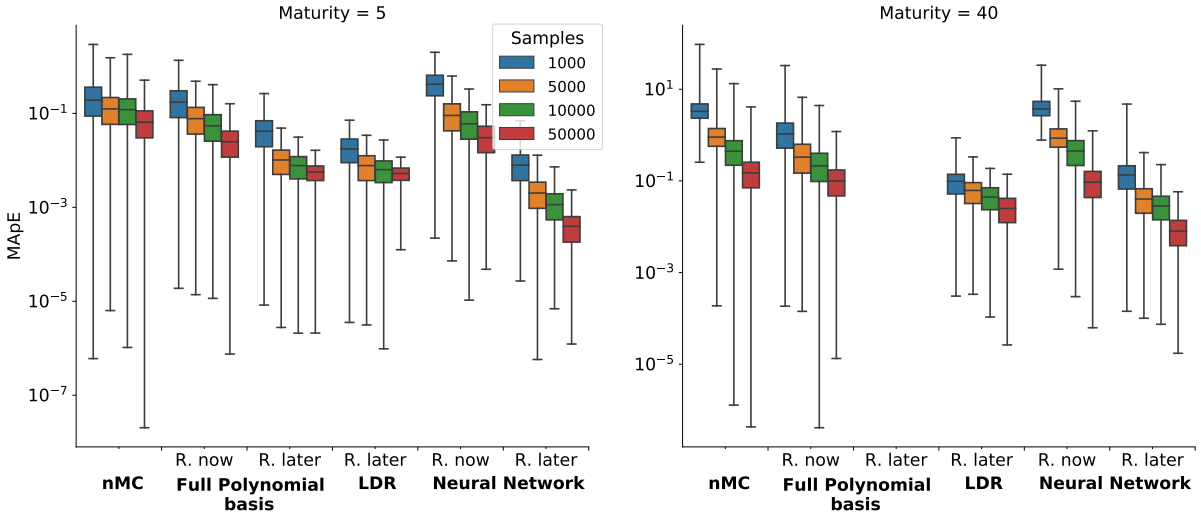


Figure 3.2: *Distribution of expected shortfall estimates per method for the European call example. Boxes show the upper and lower quartiles of the empirical distributions, while whiskers show their maxima and minima. Due to the logarithmic scale, a visual artifact is introduced by which the inter-quartile 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.*

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 (3.6), 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.

3.4.2 Comparison to lasso

The lasso (Tibshirani (1996)) 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 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 3.3.3 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 (3.17).

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 Zou et al. (2007).

Table 3.10 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 3.10: *European call, comparison of MApE ES and relative mean L_1 error (both in percentage points)*

	MApE ES			**Rel. mean L_1 error**		
Samples	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

3.4.3 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 3.11. 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 3.11: *European call, comparison of runtime (in seconds), single core AMD Opteron 6380*

		Full Polynomial basis		**LDR**	**Neural Network**	
Samples	**Lasso**	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

3.5 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 3.8 and used in previous examples. The real estate model is the same as the equity model from Section 3.8, but uses an independent stochastic driver and a lower volatility than the equity model. The stochastic mortality follows a Lee–Carter model (Lee and Carter (1992)) 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)

Bond prices (B), equity and real estate indices (EQ and RE) are simulated according to the model in Section 3.8.

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

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

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 3.8.3)

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

3.5.1 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 Ahlgrim et al. (2008) 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 Hong et al. (2017), "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 3.5.2, 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.

3.5.2 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 3.12 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 3.5, 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.

In the estimation of the expected shortfall, shown in Table 3.13 and Figure 3.3, and the analysis of the mean relative L_1 error, shown in Table 3.14, we can observe that regress-later methods dominate over regress-now methods and nested Monte Carlo, with

Table 3.12: *Insurance liability, comparison of present value MApE (in percentage points)*

		Full Polynomial basis		LDR	Neural Network	
Samples	**nMC**	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

Table 3.13: *Insurance liability, comparison of expected shortfall MApE (in percentage points)*

		Full Polynomial basis		LDR	Neural Network	
Samples	**nMC**	Regress-now	Regress-later	Regress-later	Regress-now	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

better mean absolute error and standard deviation. Unlike the case in the European call example where neural networks completely dominated the quality comparison, polynomial LDR shows better results in a few cases. However, which method shows better results is very sensitive to the choice of hyper-parameters. We provide a sensitivity analysis for hyper-parameters in Sections 3.10 and 3.11. 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-

Table 3.14: *Insurance liability, 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	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

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 3.5, 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 3.13. This behaviour is not present in the relative mean L_1 error, as evidenced in Table 3.14. 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.

3.5.3 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 3.15. Unsurprisingly, we find regress-now meth-

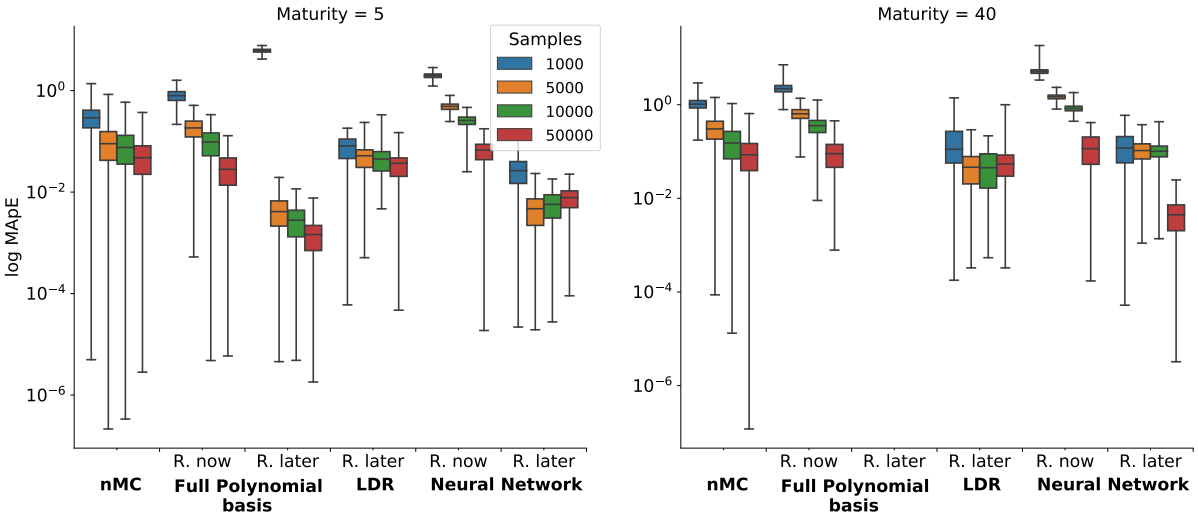


Figure 3.3: *Distribution of expected shortfall estimates per method for the insurance liability example. Boxes show the upper and lower quartiles of the empirical distributions, while whiskers show their maxima and minima. Due to the logarithmic scale, a visual artifact is introduced by which the inter-quartile 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.*

ods 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 3.15.

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

Table 3.15: *Insurance liability, comparison of runtime (in seconds), single core AMD Opteron 6380*

	Full Polynomial basis		**LDR**	**Neural Network**	
Samples	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

3.6 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

3.7 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 Willmott and Matsuura (2005) and Chai and Draxler (2014).

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

3.7.1 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 $j = 1 \dots R$, the mean absolute percentage error (MApE)

is defined as

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

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

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

3.7.2 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

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

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.

3.8 Economic scenario generator

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

3.8.1 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., Glasserman (2013). The nominal price at time t of a zero-coupon bond with maturity T is given by

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

where

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

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

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

$$\begin{aligned} g(t) &= \int_t^{t+1} e^{-\kappa(t+1-s)} b(s) ds \\ \sigma_r &= \frac{\sigma^2}{2\kappa}(1 - e^{-2\kappa}) \end{aligned}$$

and set

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

As for Y_t, define

$$\begin{aligned} h(t) &= h(t, t+1) \\ \sigma_Y &= \frac{\sigma^2}{\kappa^2}\Big(1 + \frac{1}{2\kappa}(1 - e^{-2\kappa}) + \frac{2}{\kappa}(e^{-\kappa} - 1)\Big) \\ \sigma_{rY} &= \frac{\sigma^2}{2\kappa}(1 + e^{-2\kappa} - 2e^{-\kappa}) \\ \rho_{rY} &= \sigma_{rY}/(\sigma_r \sigma_Y) \end{aligned}$$

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 $X'_{2,t+1} = \rho_{rY} X_{1,t+1} + \sqrt{1 - \rho_{rY}^2} X_{2,t+1}$.

3.8.2 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.

3.8.3 Lee–Carter parameters

The Lee–Carter parameters for the mortality model are based on the findings in the original paper Lee and Carter (1992), and they are shown in Table 3.16.

Table 3.16: *Lee–Carter parameters a_x and b_x for every age x*

x	$\mathbf{a_x}$	$\mathbf{b_x}$
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

3.9 Proofs

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

3.9.1 Proof of Lemma 1

In view of (3.1) and by orthogonality, we have $\|f - \phi_\theta^\top \beta\|_{L^2_\mathbb{Q}}^2 \geq \|f - \phi_\theta^\top \beta_\theta\|_{L^2_\mathbb{Q}}^2 = \|f\|_{L^2_\mathbb{Q}}^2 - \langle f, \phi_\theta^\top \beta_\theta \rangle_{L^2_\mathbb{Q}}$, for all $(\theta, \beta) \in \Theta \times \mathbb{R}^m$. On the other hand, by assumption (iii) we can write $\beta_\theta = \langle \phi_\theta, \phi_\theta^\top \rangle_{L^2_\mathbb{Q}}^{-1} \langle \phi_\theta, f \rangle_{L^2_\mathbb{Q}}$ and hence $\langle f, \phi_\theta^\top \beta_\theta \rangle_{L^2_\mathbb{Q}} = \langle f, \phi_\theta^\top \rangle_{L^2_\mathbb{Q}} \langle \phi_\theta, \phi_\theta^\top \rangle_{L^2_\mathbb{Q}}^{-1} \langle \phi_\theta, f \rangle_{L^2_\mathbb{Q}}$. Hence (3.5) is equivalent to (3.1) and

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

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

3.9.2 Proof of Theorem 2

Theorem 2 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) $\operatorname{span}\{\phi_{(A,b),1}, \ldots, \phi_{(A,b),m}\} = \operatorname{span}\{\phi_{(\tilde{A},\tilde{b}),1}, \ldots, \phi_{(\tilde{A},\tilde{b}),m}\}$

Proof. (i)⇔(ii): is elementary.
(ii)⇒(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)⇒(i): we argue by contradiction and assume that there exists some $\tilde{x} \in \ker \tilde{A}^\top \setminus \ker A^\top$. 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 $h \notin \operatorname{span} \phi_{(\tilde{A},\tilde{b})}$ 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. □

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)⇒(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)⇒(i): is elementary. □

3.9.3 Proof of Theorem 3

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 (3.5). 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 $\|f - \phi_A^\top \beta\|_{L^2_\mathbb{Q}} = \|f - \phi_{V^\top A}^\top \beta\|_{L^2_\mathbb{Q}}$. But $\operatorname{span}\{\phi_{A,1}, \dots, \phi_{A,m}\} \neq \operatorname{span}\{\phi_{V^\top A,1}, \dots, \phi_{V^\top A,m}\}$ in general by Theorem 2. This completes the proof of Theorem 3.

3.9.4 Proof of Theorem 4

We follow the heuristic arguments of He et al. (2020). First, note that any linear combination $h = \sum_{j=1}^m c_j \phi_{(a_j,b_j)}$ is a continuous, piece-wise affine function. As such it is *Bouligand differentiable* on $\mathbb{R}^{dT}$, see (Scholtes, 2012, Theorem 3.1.2). That is, its directional derivative $\nabla_v h(x) = \lim_{\epsilon \downarrow 0} (h(x + \epsilon v) - h(x))/\epsilon$ 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 (Scholtes, 2012, Corollary 3.1.1).

We also complete some arguments in He et al. (2020), who do not explain what kind of derivative "$\nabla \sum_{i=1}^m c_j \phi_{(a_j,b_j)}$" stands for.

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 $\phi_{(a_i,b_i)} \in \operatorname{span}\{\phi_{(\tilde{a}_1,\tilde{b}_1)}, \dots, \phi_{(\tilde{a}_m,\tilde{b}_m)}\}$, 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 $(a_i, b_i) \in \{\pm(\tilde{a}_1, \tilde{b}_1), \dots, \pm(\tilde{a}_m, \tilde{b}_m)\}$. Since i was arbitrary, we obtain $\{\pm(a_1, b_1), \dots, \pm(a_m, b_m)\} \subseteq \{\pm(\tilde{a}_1, \tilde{b}_1), \dots, \pm(\tilde{a}_m, \tilde{b}_m)\}$. 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 (3.13) does not hold, so that $\phi_{(a_i,b_i)} \in \operatorname{span}\{\phi_{(a_j,b_j)} \mid j \neq i\}$ for some i. Hence (3.12) 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 4.

Remark 1. *One may reckon that* (3.12) *and* (3.13) *together imply*

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

However, this is not true in general. Indeed, let $(a_1, b_1), \dots, (a_3, b_3) \in S_{dT}$ *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* $\operatorname{span} F = \operatorname{span} \tilde{F}$. *But* $S \neq \tilde{S}$.

3.9.5 Proof of Theorem 5

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

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

Hence $\inf_{(\theta,\beta)\in(S_1)^2\times\mathbb{R}^2} \|f - \phi_\theta^\top \beta\|_{L^2_{\mathbb{Q}}} = \lim_n \|f - \phi_{\theta_n}^\top \beta_n\|_{L^2_{\mathbb{Q}}} = 0$, but the infimum is not attained, $\|f - \phi_\theta^\top \beta\|_{L^2_{\mathbb{Q}}} > 0$ 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 $\|f - \phi_{(A,b)}^\top \beta\|_{L^2_{\mathbb{Q}}} = \|f - \phi_{(V^\top A,b)}^\top \beta\|_{L^2_{\mathbb{Q}}}$. But $\operatorname{span}\{\phi_{(A,b),1}, \ldots, \phi_{(A,b),m}\} \neq \operatorname{span}\{\phi_{(V^\top A,b),1}, \ldots, \phi_{(V^\top A,b),m}\}$ in general by Theorem 4. This completes the proof of Theorem 5.

3.10 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.

3.10.1 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 $A_0 = B \mid B_{ij} = 1 \text{ if } (i = j \wedge i \neq p) \wedge B_{ip} = \frac{1}{\sqrt{dT-p+1}} \text{ if } i \geqslant p$. 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 3.3.4:

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

Since this method, which we call *folding*, provides the best results we also use it in Sections 3.4 and 3.5. In the latter we work with $T = 5$, $d = 5$ and $p = 10$, and leads to a starting point (in block notation):

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

In Tables 3.17 and 3.18 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 3.17: *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 3.18: *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

3.10.2 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 3.19 and 3.20 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 3.19: *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 3.20: *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

3.11 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 3.5, 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 3.5. The results are summarized in Tables 3.21 and 3.22. 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 3.21: *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 3.22: *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 Hanin and Sellke (2017). In our case, it means using $p = 25$ for $T = 5$ and $p = 200$ for $T = 40$. This method does not show a good performance. Interestingly, it performs worse even for $T = 40$ where $p = 100$ is below the minimum. Still, this is not a violation of the theoretical minimum since 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.

Bibliography

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

Adelmann, M., Fernandez Arjona, L., Mayer, J., and Schmedders, K. (2019). A large-scale optimization model for replicating portfolios in the life insurance industry. *Working Paper, University of Zurich*.

Ahlgrim, K. C., D'Arcy, S. P., Gorvett, R. W., and ARM, F. (2008). A comparison of actuarial financial scenario generators. *Variance*, 2(1):111–134.

Akaike, H. (1998). Information theory and an extension of the maximum likelihood principle. In *Selected papers of hirotugu akaike*, pages 199–213. Springer.

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

Bauer, D., Bergmann, D., and Reuss, A. (2010). Solvency II and nested simulations–a least-squares Monte Carlo approach. In *Proceedings of the 2010 ICA congress*.

Bektas, S. and Sisman, Y. (2010). The comparison of L_1 and L_2 minimization methods. *International Journal of the Physical Sciences, September 2010*, 5(11):1721–1727.

Beutner, E., Pelsser, A., and Schweizer, J. (2013). Fast convergence of regress-later estimates in least squares monte carlo. *Available at SSRN 2328709*.

Beutner, E., Pelsser, A., and Schweizer, J. (2016). Theory and validation of replicating portfolios in insurance risk management. *Available at SSRN 2557368*.

Bishop, C. (2007). Pattern recognition and machine learning (information science and statistics), 1st edn. 2006. corr. 2nd printing edn. *Springer, New York*.

Bloomfield, P. and Steiger, W. (1983). Least absolute deviations: Theory, applications, and algorithms. In Huber, P. and Rosenblatt, M., editors, *Progress in Probability and Statistics*, volume 6. Birkhauser, Boston.

Boekel, P., van Delft, L., Hoshino, T., Ino, R., Reynolds, C., and Verheugen, H. (2009). Replicating portfolios; An introduction: Analysis and illustrations. *Milliman Research Report November 2009*.

Boudabsa, L. and Filipović, D. (2019). Machine learning with kernels for portfolio valuation and risk management. Swiss Finance Institute Research Paper No. 19-34. https://ssrn.com/abstract=3401539.

Boyle, P. and Hardy, M. (2003). Guaranteed annuity options. *ASTIN Bulletin: The Journal of the IAA*, 33(2):125–152.

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

Burmeister, C. and Mausser, H. (2009). Using trading restrictions in replicating portfolios. *Life & Pensions Magazine*, pages 36–40.

Burmeister, C., Mausser, H., and Romanko, O. (2010). Using trading costs to construct better replicating portfolios. *Enterprise Risk Management Symposium Monograph, Society of Actuaries, April 2010*.

Burnham, K. P. and Anderson, D. R. (2002). A practical information-theoretic approach. *Model selection and multimodel inference, 2nd ed. Springer, New York*, 2.

Cambou, M. and Filipovic, D. (2016). Replicating portfolio approach to capital calculation. *Swiss Finance Institute Research Paper No. 16-25*. Available at SSRN: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2763733.

Cambou, M. and Filipović, D. (2018). Replicating portfolio approach to capital calculation. *Finance and Stochastics*, 22(1):181–203.

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

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

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

Chen, W. and Skoglund, J. (2012a). Cashflow replication with mismatch constraints. *The Journal of Risk, Summer 2012*, 14(4):115–128.

Chen, W. and Skoglund, J. (2012b). Cashflow replication with mismatch constraints. *The Journal of Risk*, 14(4):115.

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

Daul, S. and Gutiérrez Vidal, E. (2009). Replication of insurance liabilities. *RiskMetrics Journal*, 9(1):76–96.

Delbaen, F. and Schachermayer, W. (2006). *The mathematics of arbitrage*. Springer Finance. Springer-Verlag, Berlin.

Devineau, L. and Chauvigny, M. (2011). Replicating portfolios: Calibration techniques for the calculation of the Solvency II economic capital. *Bulletin Français d'Actuariat*, 11(21):59–97.

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

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

Fabozzi, F. J. (2008). *Handbook of Finance*, volume I-III. John Wiley & Sons, Inc., Hoboken, New Jersey.

Fabozzi, F. J. and Buetow, G. W. (2008a). Interest rate swaps. In Fabozzi, F., editor, *Handbook of Finance*, volume I. John Wiley & Sons, Inc., Hoboken, New Jersey.

Fabozzi, F. J. and Buetow, G. W. (2008b). Valuing swaptions. In Fabozzi, F., editor, *Handbook of Finance*, volume III. John Wiley & Sons, Inc., Hoboken, New Jersey.

Feinstein, C. D. and Thapa, M. N. (1993). A reformulation of a mean-absolute deviation portfolio optimization model. *Management Science*, 39(12):1552–1553.

Feng, R., Cui, Z., and Li, P. (2016). Nested stochastic modeling for insurance companies. *Society of Actuaries*.

Fernandez-Arjona, L. (2019). Esg simulations and rpdb cash flows (june 2019). http://dx.doi.org/10.17632/6vvzh2w4g4.1. Mendeley Data, v1.

Fernandez-Arjona, L. (2021). A neural network model for solvency calculations in life insurance. *Annals of Actuarial Science*, page to appear.

Fernandez-Arjona, L. and Filipović, D. (2020). Benchmark and training data for replicating financial and insurance examples. Zenodo.

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

Fouque, J. P., Papanicolaou, G., and Sircar, K. R. (2000). *Derivatives in financial markets with stochastic volatility*. Cambridge University Press, New York, NY.

Friedman, J., Hastie, T., and Tibshirani, R. (2001). *The elements of statistical learning*, volume 1. Springer series in statistics New York.

Gan, G. and Lin, X. S. (2015). Valuation of large variable annuity portfolios under nested simulation: A functional data approach. *Insurance: Mathematics and Economics*, 62:138–150.

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

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

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

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

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

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

Hejazi, S. A. and Jackson, K. R. (2016). A neural network approach to efficient valuation of large portfolios of variable annuities. *Insurance: Mathematics and Economics*, 70:169–181.

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

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

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

Huang, W., Gallivan, K. A., and Absil, P.-A. (2015). A broyden class of quasi-newton methods for riemannian optimization. *SIAM Journal on Optimization*, 25(3):1660–1685.

Jacobs, B. I., Levy, K. N., and Markowitz, H. M. (2005). Portfolio optimization with factors, scenarios, and realistic short positions. *Operations Research*, 53(4):586–599.

Jacobs, B. I., Levy, K. N., and Markowitz, H. M. (2006). Trimability and fast optimization of long-short portfolios. *Financial Analysts Journal*, 62(2):36–46.

Jorion, P. (2000). Risk management lessons from long-term capital managment. *European Financial Managment*, 6(3):277–300.

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

Konno, H. and Yamazaki, H. (1991). Mean-absolute deviation portfolio optimization model and its application to Tokyo stock market. *Management Science*, 37(5):519–531.

Koursaris, A. (2011). A primer in replicating portfolios. *Barrie & Hibbert Insights July 2011*.

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

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

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

Linkedin (2020). 2020 emerging jobs report. Online. https://business.linkedin.com/content/dam/me/business/en-us/talent-solutions/emerging-jobs-report/Emerging_Jobs_Report_U.S._FINAL.pdf.

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

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

McKinney, W. (2010). Data structures for statistical computing in python. In van der Walt, S. and Millman, J., editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56.

Natolski, J. and Werner, R. (2014a). Mathematical analysis of different approaches for replicating portfolios. *European Actuarial Journal*, pages 1–25.

Natolski, J. and Werner, R. (2014b). Mathematical analysis of different approaches for replicating portfolios. *European Actuarial Journal*, 4(2):411–435.

Ninomiya, Y., Kawano, S., et al. (2016). Aic for the lasso in generalized linear models. *Electronic Journal of Statistics*, 10(2):2537–2560.

Oechslin, J., Aubry, O., Aellig, M., Kappeli, A., Bronnimann, D., Tandonnet, A., and Valois, G. (2007). Replicating embedded options. *Life & Pensions Risk*, pages 47–52.

Oliphant, T. (2006–). NumPy: A guide to NumPy. USA: Trelgol Publishing. [Online; accessed November 2019].

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011a). Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

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

Pelsser, A. (2003). Pricing and hedging guaranteed annuity options via static option replication. *Insurance: Mathematics and Economics*, 33(2):283–296.

Pelsser, A. and Schweizer, J. (2016). The difference between LSMC and replicating portfolio in insurance liability modeling. *European actuarial journal*, 6(2):441–494.

Porteous, B. T. and Tapadar, P. (2008). The impact of capital structure on economic capital and risk adjusted performance. *Astin Bulletin*, 38(01):341–380.

Rice, J. R. and White, J. S. (1964). Norms for smoothing and estimation. *SIAM Review*, 6(3):243–256.

Risk, J. and Ludkovski, M. (2018). Sequential design and spatial modeling for portfolio tail risk measurement. *SIAM Journal on Financial Mathematics*, 9(4):1137–1174.

Rudolf, M., Wolter, H.-J., and Zimmermann, H. (1997). A linear model for tracking error minimization. *Journal of Banking & Finance*, 23:85–103.

Scholtes, S. (2012). *Introduction to piecewise differentiable equations*. SpringerBriefs in Optimization. Springer, New York.

Schrager, D. (2008). Replicating portfolios for insurance liabilities. *Aenorm*, 59:57–61.

Seemann, A. (2009). Replizierende Portfolios in der Lebensversicherung. *University of Ulm, Preprint Series 2009-02*.

Sullivan, T. J. (2015). *Introduction to uncertainty quantification*, volume 63 of *Texts in Applied Mathematics*. Springer, Cham.

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

Townsend, J., Koep, N., and Weichwald, S. (2016). Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *The Journal of Machine Learning Research*, 17(1):4755–4759.

Van Der Walt, S., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22.

Vidal, E. G. and Daul, S. (2009). Replication of insurance liabilities. *RiskMetrics Journal*, 9(1).

Willmott, C. J. and Matsuura, K. (2005). 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.

Zou, H., Hastie, T., Tibshirani, R., et al. (2007). On the degrees of freedom of the lasso. *The Annals of Statistics*, 35(5):2173–2192.

Zurich Insurance Group (2017). Zurich Insurance Group, Annual report 2016.

Curriculum Vitae

Personal Details

First name, last name:	Lucio, Fernandez-Arjona Berchansky
Date of birth:	11 April 1982
Citizenship:	Spain and Argentina

Education

February 17–September 21:	PhD studies in Management and Economics, University of Zurich
March 00–December 06:	Actuario, Universidad de Buenos Aires

Professional Experience

Since November 18:	Head of Life Model Risk Management, Zurich Insurance Company
March 09–October 18:	Risk and ALM Actuary, Zurich Insurance Company
May 07–February 09:	Quantitative Analyst, Criteria Investors
July 05–May 07:	Actuarial Consultant, Watson Wyatt Argentina
September 01–July 05:	Employee, Government of the City of Buenos Aires