Machine-learning regression methods for American-style path-dependent contracts*

M. Gambara†, G. Livieri‡, A. Pallavicini§

First Version: January 31, 2023. This version: July 21, 2025

Abstract

Evaluating financial products with early-termination clauses, in particular those with path-dependent structures, is challenging. This paper focuses on Asian options, look-back options, and callable certificates. We will compare regression methods for pricing and computing sensitivities, highlighting modern machine learning techniques against traditional polynomial basis functions. Specifically, we will analyze randomized recurrent and feed-forward neural networks, along with a novel approach using signatures of the underlying price process. For option sensitivities like Delta and Gamma, we will incorporate Chebyshev interpolation. Our findings show that machine learning algorithms often match the accuracy and efficiency of traditional methods for Asian and look-back options, while randomized neural networks are best for callable certificates. Furthermore, we apply Chebyshev interpolation for Delta and Gamma calculations for the first time in Asian options and callable certificates.

JEL classification codes: C63, G13.
AMS classification codes: 65C05, 91G20, 91G60.
Keywords: Amerasian options, Look-back options, Callable certificates, Early termination, Random networks, Signature methods, Least-square Monte Carlo, Chebyshev Greeks.

*The authors report no potential competing interests. The opinions expressed in this document are solely those of the authors and do not represent in any way those of their present and past employers.

†Inait SA, Address: Av. du Tribunal-Fédéral 34, 1005, Lausanne, Switzerland. Email address: matteo.gambara@gmail.com.

‡The London School of Economics and Political Science, Department of Statistics. Address: Houghton St, London WC2A 2AE, United Kingdom. Email address: g.livieri@lse.ac.uk.

§Intesa Sanpaolo, Financial Engineering. Address: largo Mattioli 3, Milano 20121, Italy. Email address: andrea.pallavicini@intesasanpaolo.com.

Contents

List of Tables

List of Figures

1 Introduction

Financial products with early-termination features have become increasingly popular. These products allow investors to participate in the performance of underlying assets while granting them the flexibility to exit their positions before the maturity date. The investor or the issuer may have the right to terminate the contract, subject to specific conditions. These products are commonly structured as derivatives contracts or certificates. The early-termination options within these products allow for swift liquidation triggered by factors such as asset performance or specific events.

Evaluating and calculating sensitivities for financial products with early-termination clauses presents significant challenges, especially when these products have path-dependent structures. Traditional techniques, such as least-squares Monte Carlo (LSMC) simulations, can struggle with the "curse of dimensionality." LSMC relies on ordinary least-squares (OLS) approximation, which requires the selection of appropriate *basis functions*. As the dimensionality of the regression basis increases, the performance of LSMC deteriorates, growing polynomially or even exponentially with the number of risk factors, as discussed in Section 2.2 of Longstaff et al. (2001). Therefore, the selection of the basis function is crucial for the analysis in this paper.

The aim of this paper is to examine the performance of various regression methods in pricing and computing sensitivities. We will discuss the advantages and limitations of modern approaches made possible by recent advances in machine learning, which serve as alternatives to traditional polynomial basis functions. Specifically, we will enhance regression-based methods by introducing alternative basis function choices derived from randomized neural networks and signatures, and we will analyze whether this approach offers a promising solution to the challenges posed by path-dependent early-terminating financial products such as Asian options, look-back options, and callable certificates. Instead, for computing option sensitivities, particularly Delta and Gamma, this paper complements these techniques with Chebyshev interpolation.

Asian and look-back options are usually traded in over-the-counter markets, providing investors with opportunities to capitalize on the average or extreme values of underlying asset prices over a predetermined period. The time window associated with these options can either be fixed, where the average or extreme value is calculated over a specific period, or rolling, where the calculation period continuously updates as time progresses. In certain cases, these contracts may also incorporate early-termination features, allowing investors to exit the position before expiration, as described in Kao et al. (2003), Bernard et al. (2014), and Goudenège et al. (2022). In the context of commodity markets, these types of options can be regarded as exemplifying Swing contracts. Swing contracts allow the holder to adjust the quantity of the underlying asset delivered or received within a specified time frame. We refer to Thompson (1995), Barrera-Esteve et al. (2006), and Daluiso et al. (2024) for further details.

The dimensionality issue, mentioned above, is especially pronounced in the presence of path dependence. Alternative approaches explored in the literature include partial differential equation methods (e.g., Dai et al. (2010) and Federico et al. (2015)) and lattice-based techniques (e.g., Bernhart et al. (2011) and Lelong (2019)), with the latter often applied to price options on averages. However, such methods are typically constrained to settings with limited complexity, for instance, moving-average options with a monitoring window of at most ten observations (e.g., Bernhart et al. (2011) and Lelong (2019)).

Recently, the problem of pricing early-terminating products via LSMC has been addressed by replacing the traditional basis functions with neural networks and employing gradient descent instead of OLS; see, for example, Kohler et al. (2010), Becker et al. (2020), and Lapeyre et al. (2021). However, since neural networks are non-convex functions with respect to their trainable

parameters, gradient descent does not necessarily converge to a global minimum under typical training procedures (e.g., finite training time, explicit regularization). In contrast, OLS minimization of linear systems guarantees global convergence. This lack of convergence guarantees is a significant drawback of these methods, as they often rely on strong and unrealistic assumptions to establish theoretical results. Additionally, these approaches necessitate an off-line training phase that must be repeated whenever market conditions change or a different payoff structure is considered, further complicating their practical implementation.

A tentative to overcome these problems is proposed in Herrera et al. (2024), where randomized neural networks are employed as a linear regression basis. A randomized neural network can be viewed as a linear combination of random basis functions. In this approach (e.g., Cao et al. (2018)), instead of training all layers of the neural network, the parameters of the hidden layers are randomly initialized and kept fixed, while only the parameters of the output layer are optimized. This significantly accelerates the training process, enabling it to be performed on-line whenever a price is required.

In the present paper, as a first contribution, we adapt the approach of Herrera et al. (2024) to deal with the previously discussed payoffs, and we compare the results with an alternative original formulation based on signature methods. In particular, we employ both non-randomized signatures and randomized signatures. The former are novel mathematical tools that have been developed to study complex and irregular paths and functions. They represent a path in terms of its iterated integrals, hence capturing essential information about path roughness and regularity. In the present context, non-randomized signature methods offer a unique advantage as they provide a way to express the state path dependency in terms of a linear function of a limited number of risk factors so that we can apply standard regression techniques. However, they may suffer from similar disadvantages as polynomials, such as the "curse of dimensionality", which can be overcome by randomized signatures, i.e. algorithms that try to approximate input/output systems without the need of a full calibration of the system itself.

The literature on non-randomized signature methods with applications in finance is huge, so that, in this contribution, we limit ourselves to cite only the most recent research papers on the topic focusing, to the best of our knowledge, on the pricing of financial derivatives. Sabate-Vidales et al. (2020) use a combination of recurrent neural network and signature methods to design efficient algorithms for solving parametric families of path-dependent partial differential equations (PDE) that arise in pricing and hedging of path-dependent derivatives or from use of non-Markov models. A path-dependent PDE solver, based on signature kernels, is also proposed in Pannier et al. (2024). On the other hand, the pricing of exotic vanilla derivatives is covered in T. Lyons et al. (2019). Finally, methods based on signatures for pricing path-dependent options can be found also in Feng et al. (2021) and Bayraktar et al. (2022), where signatures are used as features for either feed-forward or recurrent neural networks. We claim that the use of deep neural networks here is not strictly necessary, whence the novelty of our approach when using signatures as basis functions, because of the ability of the signature by itself to represent data using a small set of features, by capturing the most important properties of data in a non-parametric way. On the other hand, examples of applications of randomized signatures can be found in Akyildirim et al. (2022), where they are used to identify pump-and-dump attempts in the crypto market in an unsupervised learning settings, and in Akyildirim et al. (2023), where randomized signatures are utilized as a non-parametric and non-linear drift estimator to find an optimal allocation a long-only portfolio. Non-randomized and randomized signature methods, along with the corresponding relevant mathematical literature, are discusses and reported in Section 3.3

As a second contribution, we then analyze algorithms to compute sensitivities, particularly Delta and Gamma. As shown in Herrera et al. (2024), second-order sensitivities are quite noisy

when regression methods are used for pricing, and they require particular care to be calculated in a robust way. We investigate the regression-based method by Létourneau et al. (2023) and the Chebyshev interpolation techniques developed in Maran et al. (2022). To the best of our knowledge, it is the first time that this analysis is performed for American-style Asian and look-back options and for callable certificates.

Our numerical results indicate that machine learning-based algorithms achieve accuracy and computational efficiency that are comparable to traditional algorithms when pricing Asian and look-back options. In contrast, we find that randomized neural networks are the most effective choice of basis functions for pricing callable certificates. Finally, we discovered that Chebyshev interpolation techniques can be successfully applied, for the first time, to Delta and Gamma calculations for both Asian options and callable certificates.

The paper is organized as follows. In Section 2 we discuss the payoffs under investigation, namely American-style Asian and look-back options. In Section 3, we present the pricing algorithms. We start by introducing the standard approach based on the LSMC; then we continue with the randomized neural networks, and the signature methods. We end the section by discussing how to compute sensitivities. In Section 5 we present the numerical details of the algorithms and we perform investigations on the different payoffs we have presented. As a last contribution we discuss the case of callable certificates in Section 6. Then, in Section 7 we summarize the paper and we hint at future developments. The paper concludes with different appendices that should deepen some theoretical aspects of the employed algorithms (Appendix A and B), the theory of other derivatives with similar features, such as Snowball and Lock-in callable certificates (Appendix C), the numerical methods of sensitivity calculations (Appendix D), and provide more numerical results on the proposed methods for the interested reader (Appendix E).

1.1 Notation

For the sake of the reader, we collect and define here *some* notations that we will use in the rest of the paper, and we indicate the exact point where the first appearance of a symbol occurs:

1. M: the length of the moving window over which the average, minimum or maximum of the price process is computed when pricing Asian and look-back payoffs; 2.1.
2. $\{T_0, \dots, T_N\}$, where $T_0 := 0$ and $T_N := T$: observation dates; 2.1.
3. V_{T_i}: value function at T_i; Equation (6); C_{T_i}: continuation value at T_i; Equation (8). B_{T_i}: bank-account process at T_i; Equation (6).
4. $\{\varphi_1, \dots, \varphi_B\}$: set of B basis functions; just below Equation (10).
5. h: hidden size of random neural networks; Subsection 3.2.
6. d: dimension of the state process at each observation date T_i; Subsections 3.2 and 3.3, and also Appendix B.
7. n: order of the truncation of the signature; Subsection 3.3 and Appendix B.
8. $k^\star$: dimension of the reservoir system in the randomized signature approach; Equation (13).
9. ς: variance of the independent normal random variables used to populate the random matrices in the randomized signature approach; Subsection 3.3.2.

10. P and Q: number of discrete paths for the price process via Gaussian Process Regression and number of points employed in the Gauss Hermite Quadrature; Subsection 5.1.

11. R^ρ with $\rho \in \{1,2,3,4\}$: sets of risk factors; 5.1.

2 Financial products with early termination

In the present paper, we discuss numerical algorithms to price path-dependent financial products with early-termination clauses. In this section, we describe the selection of payoffs considered in our main analysis: Asian and look-back options.

2.1 Asian and look-back payoffs

In many markets, ranging from equity to commodity exchanges, other-the-counter options are typically sold with early-termination clauses. Call and put options, as well as Asian and look-back options may share such clauses.

In general, we consider options up to a maturity date T written on an underlying asset, whose price process we term S_t with $t \in [0,T]$. These options are written either on the average, minimum or maximum of the price process over a moving window of M days. In order to write their payoff, we first introduce a grid of daily observation dates $\{T_0,\dots,T_N\}$, where $T_0 := 0$ and $T_N := T$, and the following random variables

$$A^{\text{avg}}_{T_i}(M) := \frac{1}{M}\sum_{j=i-M+1}^{i} S_{T_j}, \tag{1}$$

along with

$$A^{\min}_{T_i}(M) := \min_{j\in[i-M+1,i]} S_{T_j}, \quad A^{\max}_{T_i}(M) := \max_{j\in[i-M+1,i]} S_{T_j}, \tag{2}$$

defined for $M-1 \le i \le N$.

Then, we can define the option payoff we wish to discuss. We start with the fixed-strike and floating-strike put Asian payoffs, respectively given by

$$\Psi^{\text{A1}}_{T_i}(M;K) := \max\left(K - A^{\text{avg}}_{T_i}(M), 0\right), \quad \Psi^{\text{A2}}_{T_i}(M) := \max\left(S_{T_i} - A^{\text{avg}}_{T_i}(M), 0\right), \tag{3}$$

where $K > 0$ is the strike price. Moreover, we introduce the corresponding look-back payoffs, given by

$$\Psi^{\text{L1}}_{T_i}(M;K) := \max\left(K - A^{\max}_{T_i}(M), 0\right), \quad \Psi^{\text{L2}}_{T_i}(M) := \max\left(S_{T_i} - A^{\min}_{T_i}(M), 0\right). \tag{4}$$

In the following, we will use the symbol $\Psi_{T_i}(M)$ to denote them if it is clear from the context to which payoff we are referring. We observe that since we consider that the first available underlying value is S_0, the payoff function cannot be evaluated before time T_{M-1}. Our option can be exercised at any time instant T_i, for $i = M,\dots,N$. It is worth noticing that, following Bernhart et al. (2011), Lelong (2019) and Goudenège et al. (2022), the first time the option can be exercises is T_M, but other choices are possible. In the above definitions we follow Bernhart et al. (2011) and we always assume that $M-1 < i \le N$, so that we never consider payoffs with moving windows in the past.

2.2 Early exercise and early termination

The payoffs described in the previous subsection are paid by financial products which can be terminated before the contract maturity date T by the option buyer.

In this section, we setup the pricing framework for this products. We choose a risk-neutral pricing model $(\Omega, \mathcal{F}, \mathbb{Q})$, with a filtration $(\mathcal{F}_t)_{t\in[0,T]}$ representing all the observable market quantities. Thus, the price processes S_t are adapted to such a filtration.

We consider that the options can be exercised on any date in the time grid $\{T_M, \ldots, T_N\}$ previously defined. We setup the pricing problem by starting from the last exercise date T_N, and we introduce the $\mathcal{F}_{T_N}$-measurable value function V_N, defined as

$$V_{T_N} := \Psi_{T_N}, \tag{5}$$

where Ψ is one of the Asian or look-back payoffs introduced in the previous Subsection 2.1 (see Equations (3) and (4)); here we have not made explicit the dependence on the parameters M and K in Ψ, for the ease of notation. On the previous dates T_i, with $M-1 < i < N$ the option buyer may decide to exercise the option, by choosing the maximum between the immediate exercise and the continuation value of the contract, precisely defined below. Thus, we can write

$$V_{T_i} := \max\left\{\Psi_{T_i}, B_{T_i}\,\mathbb{E}\left[\frac{V_{T_{i+1}}}{B_{T_{i+1}}}\,\middle|\,\mathcal{F}_{T_i}\right]\right\}, \tag{6}$$

where B_t is the bank-account process, and the expectation is taken under the pricing measure $\mathbb{Q}$. We assume that Ψ_{T_i} is square integrable for all $M-1 < i \leq N$.

The price of the option can be calculated by applying the above recursion up to T_M, before taking the expectation under $\mathbb{Q}$.

$$V_0 := \mathbb{E}\left[\frac{V_{T_M}}{B_{T_M}}\right]. \tag{7}$$

3 Pricing techniques

This section presents the different pricing techniques we will investigate in our numerical experiments later on in Section 4. In the case of floating-strike Asian options we will use also two methods proposed in Goudenège et al. (2022). The former is based on Gaussian Process Regression and Gauss-Hermite quadrature, and it is named GPR-GHQ. The latter is based on the construction of a binomial Markov chain. It is beyond the scope of the present article to adapt their methodologies to other payoffs. We refer to Appendix A for a description of the main features of the two methodologies.

In the dynamic problem of Equation (6) a choice is made between an intrinsic value, Ψ_{T_i}, and a continuation value, the latter being defined by a conditional expectation as

$$C_{T_i} := B_{T_i}\mathbb{E}\left[\frac{V_{T_{i+1}}}{B_{T_{i+1}}}\,\middle|\,\mathcal{F}_{T_i}\right] \tag{8}$$

with $M-1 < i < N$; the continuation value is, in general, not trivial to calculate. In addition, we notice that it is not only a function of the last stock price, say S_{T_i}, but a function that potentially depends on the entire history $S_{T_M}, \ldots, S_{T_i}$. In what follows, it may happen that we will denote explicitly the dependence of the continuation value on the information set if necessary.

Now, we proceed by describing how we approximate the continuation values C_{T_i} given by Equation (8).

3.1 Least-square Monte Carlo

A standard way to approximate the continuation values given by Equation (8) is the LSMC described by Tilley (1993), Barraquand et al. (1995), and Longstaff et al. (2001). The LSMC method re-writes the dynamic problem given by Equation (6) as

$$V_{T_i} = \mathbb{1}_{\{\Psi_{T_i} > C_{T_i}\}} \Psi_{T_i} + \mathbb{1}_{\{\Psi_{T_i} < C_{T_i}\}} C_{T_i}. \tag{9}$$

First, we approximate the continuation values occurring within the indicator functions as

$$C_{T_i} \approx C^{\theta}_{T_i} := \sum_{b=1}^{B} \theta_b \varphi_b(S_{T_1}, \dots, S_{T_i}) \tag{10}$$

where $\{\varphi_1, \dots, \varphi_B\}$ is a set of B basis functions and $\theta \in \mathbb{R}^B$ are the trainable weights, which can be estimated by standard linear regression. Then, if we expand the recursion, we notice that the extant conditional expectation at time T_i is evaluated within expectations conditioned at previous times, up to time T_M. Thus, the dynamic problem can be formulated in an equivalent way, thanks to the tower rule, as

$$V_0 \approx \mathbb{E}\left[\frac{V^{\theta}_{T_M}}{B_{T_M}} \right], \quad V^{\theta}_{T_i} := \mathbb{1}_{\{\Psi_{T_i} > C^{\theta}_{T_i}\}} \Psi_{T_i} + \mathbb{1}_{\{\Psi_{T_i} < C^{\theta}_{T_i}\}} \frac{B_{T_i}}{B_{T_{i+1}}} V^{\theta}_{i+1}, \quad V^{\theta}_{T_N} := \Psi_{T_N}. \tag{11}$$

We warn the reader that $V^{\theta}_{T_i}$ does not represent the option value at time T_i since it depends on future realization of the price process. The option value is its expectation conditioned at time T_i.

Different types of basis functions have been proposed in the literature so far. In the original paper of Longstaff et al. (2001), the authors discuss Laguerre, Hermite, Legendre, Chebyshev, Gegenbauer, and Jacobi polynomials. A Laguerre polynomial approximation is used also in Bernhart et al. (2011) to price a continuously monitored moving average options. In his Ph.D. thesis, Grau (2008) employs a sparse polynomial basis for the regression in order to attenuate the numerical inefficiency of the LSMC method. However, the latter does not easily scale to high-dimensional problems. Dirnstorfer et al. (2013) use sparse grid basis functions based on polynomials or piecewise linear functions. Lelong (2019) replaces the standard least-square regression with a Wiener chaos expansion. Finally, we mention the Ph.D. thesis of Bilger (2003), where polynomials of the underlying index and the average are employed. We will see in Section 4 that the choice of the basis functions will have a non-negligible impact on the pricing performances.

3.2 Randomized neural networks

We employ two types of approximations of the continuation value based on randomized neural networks. The use of randomized neural networks for computing the price of American options has been firstly introduced in Herrera et al. (2024), with impressive results in terms of speed and accuracy.

In fact, this method allows to build a random basis that is rich enough for a good fit at a cheap computational cost, and just train the last layer for the network, which amounts to just solve a linear regression (at most with regularization). In this way, randomized neural networks can be viewed as an alternative to the polynomial bases, usually adopted when implementing the LSMC method, which can better represent non-linearity and path-dependency of the continuation value. The first type of randomized neural network approximates the continuation function via a randomized feed-forward neural network; we call this type of networks R-FFNN. On the other

hand, the second type employs a randomized recurrent neural network for the approximation of the continuation value; we call this type of network R-RNN.

In the upcoming subsections, we follow the presentation by Herrera et al. (2024) for the description of the two methodologies; in particular we present them with a fully-connected shallow neural network.

3.2.1 R-FFNN

We start by introducing the activation function $\sigma : \mathbb{R} \to \mathbb{R}$ acting on each component of the network, for instance a rectified linear unit (ReLU) or an hyperbolic tangent (tanh) activation function. Let $\widetilde{M}, h \in \mathbb{N}_{>0}$, we define $\boldsymbol{\sigma} : \mathbb{R}^{h-1} \to \mathbb{R}^{h-1}$ as the activation function acting component-wise, in the sense that $\boldsymbol{\sigma}(x) = (\sigma(x_1), \dots, \sigma(x_{h-1}))^\top$, where $x \in \mathbb{R}^{h-1}$. The natural number $\widetilde{M}$ depends on the choice of the information set used for the construction of the random basis. For instance, in case of Asian options (see Subsection 2.1), one possibility is to use the values of the stock price over the time window of length M; in this case $\widetilde{M} = M$.

Then, let $d \in \mathbb{N}_{\geq 1}$ be the dimension of of the state process at each observation date, we introduce the parameters of the hidden layer $\vartheta := (A, b) \in \mathbb{R}^{(h-1)\times(d\widetilde{M})} \times \mathbb{R}^{h-1}$, namely the weight matrix A and the bias vector b, whose elements are randomly and identically sampled and not optimized in any training procedure (we will specify their distribution at the beginning of Section 4). Furthermore, we introduce the function $\phi : \mathbb{R}^{d\widetilde{M}} \to \mathbb{R}^h$, $x \mapsto \phi(x) = (\boldsymbol{\sigma}(Ax + b)^\top, 1)^\top$ and the parameters $\theta_i := ((A_i)^\top, b_i) \in \mathbb{R}^{h-1} \times \mathbb{R}$ to be optimized. We notice that the element 1 in ϕ which is the constant element that is added for standard linear regressions.

Finally, we are able to formulate the approximation of the the continuation value. We can write for each i

$$C_i^\theta(x) := \theta_i^\top \phi(x) = A_i^\top \boldsymbol{\sigma}(Ax + b) + b_i.$$

where θ_i can be estimated via OLS because the approximation function C_i^θ is linear in the parameters θ_i; see Theorems 5 and 6 of Herrera et al. (2024).

We can exemplify the above construction by analyzing a specific case. For instance, for the sake of exposition only, we suppose that $M = 1$, and that we have access to m realizations of the stock price paths, where the ℓ-th realization is denoted by $S_0, S_{T_1}^{(\ell)}, S_{T_2}^{(\ell)}, \dots, S_{T_N}^{(\ell)}$ with the fixed initial value S_0. Then, the parameters θ_i of the last layer are found by minimizing the squared error of the difference between conditional expectation of the discounted future price of the financial derivative, say $\frac{B_{T_i}}{B_{T_{i+1}}} V_{T_{i+1}}^{\theta,(\ell)}$ and the approximation of the continuation value evaluated at $S_{T_i}^{(\ell)}$. The minimizer has the following closed form expression:

$$\theta_i = \frac{B_{T_i}}{B_{T_{i+1}}} \left(\sum_{\ell=1}^{N_{MC}} \phi\left(S_{T_i}^{(\ell)}\right) \phi^T\left(S_{T_i}^{(\ell)}\right) \right)^{-1} \cdot \left(\sum_{\ell=1}^{N_{MC}} \phi\left(S_{T_i}^{(\ell)}\right) V_{T_{i+1}}^{\theta,(\ell)} \right)$$

The fact that the training of the previous model (more generally, of all the models described in the present paper) can be performed in a single analytical step, that is implemented efficiently in most linear algebra libraries, is the major advantage of these methods. The computational complexity of the least square method is mostly influenced by the $h \times h$ matrix inversion in the previous equation.

One limitation of the R-FFNN is that all the points in the information set have the same relevance, in the sense that the fact that the input may be a stream of data is not captured by the R-FFNN. The R-RNN in the next paragraph takes into account this information.

3.2.2 R-RNN

In the case of a R-RNN the parameters of the hidden layer, which are randomly and identically sampled and not optimized, are given by $\vartheta := (A_x, A_\xi, b) \in \mathbb{R}^{(h-1)\times(d\widetilde{M})} \times \mathbb{R}^{(h-1)\times(h-1)} \times \mathbb{R}^{(h-1)}$. We notice that, in this case, the tuning parameters are more important, as they determine the interplay between past and new information.

We start by defining $\phi : \mathbb{R}^{d\widetilde{M}} \times \mathbb{R}^h \to \mathbb{R}^{h+1}$, $(x, \xi) \mapsto \phi(x, \xi) = (\boldsymbol{\sigma}(A_x x + A_\xi \xi + b)^\top, 1)^\top$, and $\theta_i := ((A_i)^\top, b_i) \in \mathbb{R}^{h-1} \times \mathbb{R}$ the parameters to be optimized. For each i the continuation value is approximated by

$$\begin{cases} \xi_i = \boldsymbol{\sigma}(A_x x_i + A_i \xi_{i-1} + b), \\ C_i^\theta(x) = \theta_i^\top \phi(x) = A_i^\top \boldsymbol{\sigma}(Ax + b) + b_i = \theta_i^\top \phi(x_i, \xi_{i-1}), \end{cases}$$

where $\xi_{-1} = 0$. As for the R-FFNN the parameters θ_i are estimated via OLS, and so the computational complexity of the algorithm is mostly influenced by the inversion of a $(h-1) \times (h-1)$ matrix.

There are a number of hyper-parameters for both R-FFNN and R-RNN that can be tuned, namely the size of the neural network hidden state, the number of layers in the neural network, the activation function, and the distribution of the parameters of the hidden layer. For the R-RNN, we consider one layer in time for every fixing date. From a practical point of view, the main difference between R-FFNN and R-RNN is that for the former we construct a new neural network in which the number of layers coincides with the number of fixing dates, whereas for the R-RNN we construct a unique (random) neural network in which we train each time only the readout map. In particular, the dimension of the R-RNN decreases going backward in time.

3.3 Signature methods

We propose a novel alternative algorithm to R-FFNN and R-RNN based on signatures. We design the algorithm to preserve the linear regression on which is based the LSMC method, so that we can consider also the following algorithm as an alternative definition of the basis function. More precisely, we analyze two versions of the algorithm: the former one based on the truncated signature of a suitable transformation of the time series of observed asset prices, the second one on the randomized signature. Non-randomized signatures are, roughly speaking, a tool that allows to extract features and summarize a stream of information over a time interval. However, they may suffer from similar disadvantages as polynomials, such as the "curse of dimensionality", which can be overcome by randomized signatures.

3.3.1 Truncated signature

The signature process was first studied by Chen (1957, 1977), and plays a prominent role in rough-path theory introduced in the T. J. Lyons (1998). In an effort to keep the paper as self-contained as possible we provide for convenience in Appendix B, Section B.1, a self-supporting although minimalist account of key concepts and results on signatures. Here, instead, we give some results on signatures, related to the present work, within a univariate time series framework; they can be easily generalized to a multi-dimensional time series. The material is based on Levin et al. (2013), Section 4, and Gyurkó et al. (2013).

For any $0 \le i_1 < i_2 \le N$, and $i_2, i_1 \in \mathbb{N}$, in general the signature of $\{(T_j, S_{T_j})\}_{j=i_1}^{i_2}$ is computed via the following two steps:

(i) embed a time series $\{(T_j, S_{T_j})\}_{j=i_1}^{i_2}$ into a continuous path R;

(ii) compute the signature of this transformed continuous path R.

We add time as one of the variables in the time series $\{(T_j, S_{T_j})\}_{j=0}^N$ to preserve the temporal structure; this also ensures that signatures are unique. This procedure is known as time augmentation in Hambly et al. (2010) (see Section B.2 for more details).

There are several choices for embedding a time series into a continuous path. For instance, one can rely directly on a simple piece-wise linear interpolation technique. However, the signature is an infinite sequence, whence in practice some finite collection of terms must be selected. Because the magnitude of the terms exhibit factorial decay, it is usual, as in T. Lyons (2014), to simply choose the first n terms of the sequence and consider the so-called *truncated signature* of depth n. Yet, if the function to be learned depends non-trivially on the higher degree terms, then crucial information has nonetheless been lost. A remedy might be using embedding techniques which, roughly speaking, apply a point-wise augmentation to the original stream of data before taking the signature. Then, the first n terms of the signature may better encode the necessary information.

On a first glance, the embedding transformation could be seen as not strictly necessary, since it duplicates the (spatial) dimension of the time-series. In order to answer this question, in our numerical Section 4, we compare the performance of the signature method with and without embedding transformations.

In the present paper, we employ, sequentially, two embedding techniques that we define in the following: (a) the Hoff lead-lag transformation (Hoff (2006) and Flint et al. (2016)), and (b) the time-joined path.

Definition 1 (Hoff lead-lag transformation)**.** Let $\{(T_j, S_{T_j})\}_{j=0}^N$ be a univariate time series. Let $Z : [0, 2N] \to \mathbb{R}^+ \times \mathbb{R}$ be a 2-dimensional lead-lag transformation of $\{(T_j, S_{T_j})\}_{j=0}^N$, which is defined as follows:

$$Z(t) := \begin{cases} S_{T_j} e_1 + S_{T_{j+1}} e_2 & \text{if } t \in [2j, 2j+1) \\ S_{T_j} e_1 + [S_{T_{j+1}} + 2(t-(2j+1))(S_{T_{j+2}} - S_{T_{j+1}})] & \text{if } t \in [2j+1, 2j+\frac{3}{2}) \\ [S_{T_j} + 2\left(t - \left(2i+\frac{3}{2}\right)\right)\left(S_{T_{j+1}} - S_{T_j}\right)]e_1 + S_{T_{j+2}} e_2 & \text{if } t \in [2j+\frac{3}{2}, 2j+2) \end{cases} \tag{12}$$

for $t \in [0, 2N]$, and where $\{e_i\}_{i=1,2}$ is an orthonormal basis of $\mathbb{R}^2$.

Definition 2 (Time-joined transformation)**.** Let $\{(T_j, S_{T_j})\}_{j=0}^N$ be a univariate time series. Let $R : [0, 2N+1] \to \mathbb{R}^+ \times \mathbb{R}$ be a 2-dimensional time-joining path of $\{(T_j, S_{T_j})\}_{j=0}^N$, which is defined as follows:

$$R(t) := \begin{cases} T_0 e_1 + S_{T_0} t e_2, & \text{if } t \in [0, 1); \\ [T_j + (T_{j+1} - T_j)(t - 2j - 1)]e_1 + S_{T_j} e_2 & \text{if } t \in [2j+1, 2j+2); \\ T_{j+1} e_1 + [S_{T_j} + (S_{T_{j+1}} - S_{T_j})(t - 2j - 2)]e_2, & \text{if } t \in [2j+2, 2j+3); \end{cases}$$

where $0 \le j \le N-1$, $t \in [0, 2N+1]$ and $\{e_i\}_{i=1,2}$ is an orthonormal basis of $\mathbb{R}^2$.

In particular, the signature of $\{(T_j, S_{T_j})\}_{j=0}^N$ is defined to be the signature of its time-joined transformation $R(t)$. There also exists a more intuitive and straightforward definition of a lead-lag transformation. Yet, we decide to consider the so-called Hoff lead-lag transformation because its good performance has been already noticed in the literature across different data sets and algorithms; see, for instance, Fermanian (2021).

An advantage of the lead-lag transformation is that one can read the volatility of the path directly from the second term of the signature; see Remark 4.1 of Levin et al. (2013). On the other hand, the time-joined transformation, is such that the resulting continuous path exhibits a

causal dependence on the data, being an hybrid between a linear and a rectilinear interpolation; see Moor et al. (2020).

In the present work, we use truncated signatures as non-linear basis on which we can estimate linear regressions to have theoretically sounded results (see Theorem B.5). In particular, for each realization of the stock price over the time window $T_0, \dots, T_N$, we first apply, sequentially, the lead-lag transformation and the time-joined transformation to $\{(T_j, S_{T_j})\}_{j=T_{i_1}}^{T_{i_2}}$, then we apply the truncated signature of order n to the latter. We notice that by performing such transformations, every transformed path is enriched with other dimensions: the lead-lag transform will double the dimension of the original time series, while time-addition will increase it by one. In particular, the previous two transformations, applied in sequence, transform a univariate time series in a $d = 3$ dimensional time series. Only afterwards, one computes the signature. The output of the truncated signature is a vector of dimension $s_d(n) = \frac{d^{n+1}-1}{d-1}$; see Appendix B. We use this vector, by eventually enlarging it with, for instance, the average of the stock price over the considered time window, as linear regression basis for the approximation of the continuation value; a regularization can be applied to the linear regression.

The approach explained in this section for American-style Asian options has been used in Feng et al. (2021) and Bayraktar et al. (2022), but, in this case, the signature transformation is applied to the path and then fed to a feed-forward neural network, which should not be strictly necessary.

The drawback of using the truncated signature of order n is that for a sample consisting of N points in $\mathbb{R}^d$ it has an $\mathcal{O}(Nd^n)$ computational complexity, which becomes intractable for high-dimensional systems and/or for large values of n aimed at encoding a richer information of the function to be learned; the complexity is, however, linear in the number of sampled points. In order to cope with the just-mentioned issue, instead of calculating the signature of a time series, one can extract a new quantity, called randomized signature, which is easier to compute and inherits the expressiveness of the signature. This is the content of the next subsection.

3.3.2 Randomized signature

Randomized signature has been proposed as a flexible and easily implementable alternative to the well-established path signature. Despite the name, the concept of randomized signature is strictly connected to that of reservoir computing, an area of machine learning where random, possibly recurrent neural networks are used to efficiently construct regression basis on path space. In what follow, we give a brief overview of reservoir based on the randomized path signature; we first explain the notion of reservoir system, then we present signatures as reservoirs.

Randomized signatures emerged by considering signatures as reservoir systems, that is algorithms that try to approximate input/output systems without the need of a full calibration of the system itself; see, for more details, Cuchiero et al. (2022) and the references therein. A universal approximation property is granted also in this case. This intuition was first developed in Cuchiero et al. (2022) and then in Cuchiero et al. (2021), obtaining what the authors called "randomized signatures", where the approximation property relies on the Johnson-Lindenstrauss Lemma (see Johnson et al. (1984)), consisting in a random projection of the system on a space with smaller dimension that is able to preserve its geometric structure: distortion will be at most $1+\epsilon_r$ for arbitrary ϵ_r, and the embedded space will also have dimension that depends on the same ϵ_r.

One notable drawback of considering signatures as reservoirs is that the computation of the iterated integrals needed to form the basis of the path is rather lengthy and computationally intense. This problem becomes even more relevant in high dimensions or for very long time series. It is indeed in such cases that randomized signatures, leveraging a random construction,

can make a difference. The idea behind randomized signature is compressing information of the signature through the Johnson-Lindenstrauss (JL) Lemma and consider a dynamical system for this projected signature-transformed stream.

Lemma 3.1 (Johnson et al. (1984))**.** *For every $0 < \epsilon_r < 1$, an N point set Q in some arbitrary (scalar product) space $\mathbb{H} \subseteq \mathbb{R}^d$ can be embedded into a metric space $\mathbb{R}^{k^\star}$, where $k^\star = \frac{24\log(N)}{3\epsilon_r^2 - 2\epsilon_r^3} = \mathcal{O}(\frac{\log N}{\epsilon_r^2})$ in an almost isometric manner, i.e. there is a linear map $g : \mathbb{H} \to \mathbb{R}^{k^\star}$ such that $(1-\epsilon_r)\|p_1 - p_2\|^2 \leq \|g(p_1) - g(p_2)\|^2 \leq (1+\epsilon_r)\|p_1 - p_2\|^2$ for any couple of points p_1 and p_2 in Q. The map g can be chosen to be random.*

We have the following definition (see, e.g., Compagnoni et al. (2023)):

Definition 3 (Randomized signature)**.** Given $k^\star \in \mathbb{N}$ and random matrices $A_1, \ldots, A_d \in \mathbb{R}^{k^\star \times k^\star}$, random shifts $b_1, \ldots, b_d$ in $\mathbb{R}^{k^\star}$, and any fixed activation function σ, the randomized signature of a continuous and piece-wise smooth path $X : [0,T] \to \mathbb{R}^{d+1}$ is the solution of the following differential equation

$$d\mathrm{RS}(X)_t = \sum_{i=0}^{d} \boldsymbol{\sigma}(A_i \mathrm{RS}(X)_t + b_i)\, dX_t^i, \quad \mathrm{RS}(X)_0 \in \mathbb{R}^{k^\star}. \tag{13}$$

Notice that we typically take $X_t^0 = t$.

In particular, A_i and b_i can be sampled from a normal distribution and are fixed once for all before applying the Euler scheme to Equation (13). More precisely, once that a dimension $k^\star \in \mathbb{N}$ is fixed, every entry of the random matrices can be chosen independently from a normal distribution $\mathcal{N}(0, \varsigma^2)$. We also notice that the function $\boldsymbol{\sigma}$, which might remind the reader to the activation function of neural networks, does not need to be non-linear and can be chosen to be the identity. Most importantly, the activation function needs to be injective to obtain a non-singular $\boldsymbol{\sigma}(A_i \mathrm{RS}(X)_t + b_i)$ for any $i = 1, \ldots, d$. For a discussion on the possible assumptions on the activation functions, we refer the reader to Akyildirim et al. (2022) and Biagini et al. (2024). This is not the only viable definition of randomized signatures based on the JL lemma. Another possible definition is present in the work by Cuchiero and Möller (Cuchiero et al. (2022), Definition 3.8) where the distinction between the application of randomized linear functionals from the JL lemma to truncated signature coefficients is seen as an alternative to considering randomized signatures as solution to Equation (13). The former are thus called *JL-signatures*, while the latter *randomized signatures*. The definition we give here is more in line with the reservoir systems' literature outlined in previous paragraphs.

Examples of applications of randomized signatures can be found, for example, in Akyildirim et al. (2022), where they are used to identify pump and dump attempts in the crypto market in an unsupervised learning settings, and in Akyildirim et al. (2023), where randomized signatures are utilized as a non-parametric and non-linear drift estimator to find an optimal allocation a long-only portfolio.

The computational complexity for calculating RS in Equation (13) is $\mathcal{O}(k^{\star 2} d)$ and at any time its dimension is $\mathcal{O}(k^\star)$ (when solving Equation (13) numerically, the dimension is also proportional to the number of time steps used for the Euler scheme). In practice, $k^\star \in \mathbb{N}$ is normally chosen as a parameter of the method and its value must be commensurate with the expressiveness we require from the algorithm. In Section 4 we show experimentally that, in order to match the pricing capabilities of the truncated signature of order n, the number $k^\star$ of required randomized signatures is fairly small, thus leading to a dimension that is way smaller than the dimension of the standard signature $s_d(n)$ introduced in the previous section. This confirms that working with randomized signatures is often less computationally demanding.

3.4 Sensitivity computation

The pricing algorithm we are discussing for early-termination products require the computation of a regression at each time step of the simulation algorithm. This calculation introduces additional noise in the results, so that the numerical evaluation of sensitivities to market parameters could be unstable. In particular, this can be the case for second-order sensitivities.

We consider the case of calculating first and second order sensitivities with respect to the initial spot price (namely we are interested in the so-called Delta and Gamma), and we start to analyze the simpler case of American put options, since in this case we have the possibility to check our methods against a standard (and reliable) implementation based on a binomial tree. We start by discussing the problems of finite-difference schemes, then we introduce two alternative methods: (i) the regression method proposed in Létourneau et al. (2023), and later used in Herrera et al. (2024), and (ii) the proposal of Maran et al. (2022) based on Chebyshev interpolation. We extend the latter on to our specific payoff case.

3.4.1 Finite-difference method

The simplest, and more direct, approach for sensitivities is the discretization of derivative operators in term of finite differences. If we consider a positive number ϵ, we can approximate Delta and Gamma as given by

$$\frac{\partial V}{\partial S} \approx \frac{V(S(1+\epsilon)) - V(S(1-\epsilon))}{2\epsilon}\,, \quad \frac{\partial^2 V}{\partial S^2} \approx \frac{V(S(1+\epsilon)) - V(S) + V(S(1-\epsilon))}{\epsilon^2} \tag{14}$$

for small values of ϵ, where $V(s)$ is the option price calculated with spot price equal to s.

3.4.2 Regression method

We briefly describe the regression method applied to our case, but we address the reader to the original paper by Létourneau et al. (2023) for more details and for a more general exposition. We assume that for a given positive ϵ the initial spot price S_ϵ of the Monte Carlo simulation is distributed according to a normal random variable $\mathcal{N}(S, \epsilon^2)$, where S is the market spot price, so that the derivative price can be written as

$$V = \mathbb{E}[\,V(S_\epsilon)\,] \approx \frac{1}{N_s}\sum_{k=1}^{N_s} \hat{V}(S_\epsilon^{(k)})\,, \quad S_\epsilon \sim \mathcal{N}(S, \epsilon^2), \tag{15}$$

where N_s is the number of simulation paths and $S_\epsilon^{(k)}$ is the k-th extraction of the random variable S_ϵ. Such extraction is used as initial spot price for the Monte Carlo approximation $\hat{V}(S_\epsilon^{(k)})$ of the derivative price made with N_s simulation paths.

The previous decomposition allows us not only to calculate the derivative price, but also its sensitivities. Indeed, we can consider the Taylor's approximation

$$\hat{V}(x) \approx \sum_{b=0}^{B} \beta_b^{(\epsilon,B)} (x - S)^b, \tag{16}$$

where the coefficients β can be calculated by means of an OLS algorithm on the data set $\{(S_\epsilon^{(k)}, \hat{V}(S_\epsilon^{(k)}))\}_{k=1}^{N_s}$. Then, the sensitivities can be calculated as given by

$$\frac{\partial V}{\partial S} \approx \beta_1^{(\epsilon,B)}\,, \quad \frac{\partial^2 V}{\partial S^2} \approx 2\beta_2^{(\epsilon,B)} \tag{17}$$

3.4.3 Chebyshev method

The method of Maran et al. (2022), hereafter termed Chebyshev method, consists in approximating the option price, view as a function of the spot price, by means of a Chebyshev interpolator. Then, sensitivities are calculated by taking the derivative of the interpolator with respect to the spot price. The technique relies on the fact that for analytical functions the rate of convergence of the derivatives of the interpolator to the original derivatives is exponential. The strength of the method relies on the fact that, thanks to the previous property, we can increase the size of the interpolation interval to stabilize the sensitivity computation without introducing additional biases. More details on using Chebyshev interpolation techniques in option pricing can be found in Gaß et al. (2016).

Here, we present our procedure in details. For the sake of clarity we skip the technical part of dealing with Gamma discontinuities, which we will analyze in Appendix D. For a given positive ϵ we select a number N_c of spot prices $\{S_\epsilon^0, \dots, S_\epsilon^{N_c-1}\}$ where we evaluate the corresponding Monte Carlo approximations $\{\hat{V}(S_\epsilon^0), \dots, \hat{V}(S_\epsilon^{N_c-1})\}$ of the derivative price. Such spot prices can be defined as

$$S_\epsilon^\ell := S\left(1 + \epsilon \cos\left(\frac{\ell\pi}{N_c - 1}\right)\right), \tag{18}$$

where S is the market spot price. The set of points $\{(S_\epsilon^\ell, \hat{V}(S_\epsilon^\ell))\}_{\ell=0}^{N_c-1}$ will be used to train a polynomial interpolator given by

$$\hat{V}_\epsilon(S) := \sum_{\ell=0}^{N_c-1} \hat{V}(S_\epsilon^\ell) \prod_{\jmath \neq \ell}^{N_c-1} \frac{S - S_\epsilon^\jmath}{S_\epsilon^\ell - S_\epsilon^\jmath}, \tag{19}$$

The interpolator allows us not only to calculate the derivative price, but also its sensitivities by explicit calculation of the derivatives with respect to the spot price.

3.4.4 Algorithm comparison for American put Gamma

We wish to assess the performance of these methods on early-termination products before proceeding with their use in the in the numerical Section 4. We consider as a typical playground the calculation of the second derivative with respect to the spot price (Gamma) for American put options. All the calculations are made according to the Black and Scholes model, later described in Section 4.1.

Each method to compute the sensitivities may require a different number of Monte Carlo simulations; for instance, as shown above, finite differences require three different simulations to compute Gamma. Thus, in order to compare the resulting sensitivities, keeping fixed the computational time, we consider the same number of simulated paths for each approach, irrespectively of how they are distributed among the required simulations.

We list in Table 1 the results obtained for Gamma computation for the three methods for three different choices the spot price, while keeping $K = 100$ in all calculations. We compare these results with the Gamma calculations made by means of a binomial tree. We can see, as the number of simulation paths increases, that the regression method quickly deteriorates by showing a significative bias. The other two methods perform with greater accuracy. In particular, the Chebyshev method shows a smaller bias.

These results suggest us to discard the regression method, and focus only on finite-difference and Chebyshev methods, in the numerical investigations we are going to illustrate in the following sections.

American put option: Gamma									
	$\Gamma = 0.0235$, $S_0 = 85$			$\Gamma = 0.0308$, $S_0 = 100$			$\Gamma = 0.0131$, $S_0 = 115$		
MC Path	**F.Diff.**	**Regr.**	**Cheb.**	**F.Diff.**	**Regr.**	**Cheb.**	**F.Diff.**	**Regr.**	**Cheb.**
$1.5 \cdot 10^3$	0.0188	0.0224	0.0192	0.0306	0.0296	0.0310	0.0121	0.0126	0.0143
$7.5 \cdot 10^3$	0.0246	0.0250	0.0233	0.0298	0.0303	0.0325	0.0133	0.0117	0.0128
$1.5 \cdot 10^4$	0.0247	0.0243	0.0254	0.0304	0.0302	0.0340	0.0130	0.0121	0.0127
$7.5 \cdot 10^4$	0.0240	0.0241	0.0249	0.0299	0.0295	0.0310	0.0131	0.0121	0.0132
$1.5 \cdot 10^5$	0.0247	0.0242	0.0233	0.0300	0.0293	0.0300	0.0133	0.0122	0.0131
$7.5 \cdot 10^5$	0.0241	0.0241	0.0236	0.0301	0.0294	0.0304	0.0132	0.0123	0.0131
$1.5 \cdot 10^6$	0.0239	0.0241	0.0236	0.0299	0.0294	0.0305	0.0132	0.0122	0.0130

Table 1: Gamma for an American put option with maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. The first column is the total number of simulation paths used by each algorithm. Then, there are three groups, each of three columns, for different spot prices S_0. In each group, the first column is the result of the finite-difference method ($\epsilon = 1/16$), the second one of the regression method ($\epsilon = 1/8$, $B = 9$) and the third one of the Chebyshev method ($\epsilon = 1/8$). In the second line Γ is the result of the binomial tree.

4 Numerical techniques

This section describe the ingredients common to all the payoffs and algorithms. In particular, we discuss the underlying asset dynamics, the input features of the regression basis functions, and the configuration of the random networks and signature methods.

4.1 Price dynamics

Since our contribution is focused on analyzing the performance of selecting different basis functions in the LSMC algorithm, and how this selection depends on the specific payoff we consider, we think that assuming a simple setting for asset price dynamics is preferable. In particular, we evaluate the different algorithms by assuming a Black-Scholes dynamics for underlying assets. Admittedly, we could have used more complicated dynamics, such as stochastic volatility models or rough volatility models. Moreover, we only consider synthetic data, and not real market data. However, these extensions would not add to the present work's conceptual advancements

The Black-Scholes model (see Black et al. (1973)) is characterized by the following stochastic differential equation:

$$dS_t = (r - q)S_t\,dt + \sigma S_t\,dW_t, \tag{20}$$

where $t \in [0, T]$, $S_0 = s_0$, and $(W_t)_{t\geq 0}$ is a standard Brownian motion. Besides, r is the risk-free rate, q is the dividend rate, and σ is the volatility, and we assume that all these three quantities are deterministic constants.

In our numerical investigation we use the following values: $S_0 = 100$, $r = 0.05$, $q = 0$, $\sigma = 0.3$. Such values are used also in Bernhart et al. (2011) and (later on) in Goudenège et al. (2022).

4.2 Risk factors and features

In the following, we compare four different types of basis functions for the LSMC algorithm: (i) polynomials, (ii) R-FFNN, (iii) R-RNN, (iv) signatures (either randomized or not).

As shown in Equation (10) the notation $\varphi_b(S_{T_1}, \dots, S_{T_i})$, with $1 \leq b \leq B$, is used to state that basis functions can depend on any value of the price process up to regression time $T_i < T_N$. However, depending on the payoff under consideration and on the choice of the basis functions, we can refine this statement.

In our case of Asian and look-back payoffs, we should consider at each regression time T_i the observations of the underlying asset on all the previous $M-1$ days, where M is the number of days of the moving window. Thus, at each regression time T_i we should consider basis functions of the form $\varphi_b(S_{T_{i-M+1}}, \dots, S_{T_i})$. This can be simply accommodated in the polynomial case by considering monomials within a given degree in the arguments. Yet, these methods have in principle the limitation that they assign the same relevance to all the points in the information set, in the sense that they do not capture, for instance, the fact that they are a stream of data, namely there is temporal dependence among the observation of the price process. On the other hand, R-RNN and signature methods take into account this fact.

In the case of polynomial and R-FFNN basis functions when the length M of the moving window increases the performance of the algorithm deteriorates, since we need either to increase the number of basis functions or of features of the network. As a possible solution to this problem we can reduce the number of observations needed to characterize the input and we can check empirically the impact on options prices.

With this considerations in mind we proceed as following for the polynomial and R-FFNN cases. As denoted above, let M be the number of observed underlying values included in the window-average (specific to the option derivative) and T_i the evaluation time of the regression. We build our basis functions by starting from four different choices of sets of risk factors, which we denote R^ρ with $\rho \in \{1, 2, 3, 4\}$. Each risk factor set contains F_i^ρ random variables according to the following rules.

- $[R^1]$ We consider only one risk factor, namely the value of the stock price S_{T_i} at time T_i. This choice is a strong approximation, since it discards all path-dependency effect. On the other way, is the maximum speed up we can obtain.

- $[R^2]$ We consider two risk factors (only for $M > 1$, in case $M = 1$ we revert to R^1): the value of the stock price S_{T_i} at time T_i and the arithmetic moving average computed over the interval $[T_{i-M+1}, T_i]$. Also this choice is a strong approximation, but it tries to synthesize all the path-dependency effects in a single risk factor, by preserving a good speed up of the algorithm.

- $[R^3]$ We consider $M-1$ risk factors (only for $M > 1$, in case $M = 1$ we revert to R^1): the values of the stock price $S_{T_{i-M+2}}, \dots, S_{T_i}$. We do not consider S_{T_i-M+1} because the continuation value does not depend on it, since it does not contribute to the future averages. This should be the target choice if we want to consider all the relevant risk factors without introducing any approximation.

- $[R^4]$ We consider n risk factors: the value of the stock price from the first available date up to time T_i. This choice considers any risk factor, not only what is required. We introduce this choice for the risk factors to understand how the LSMC method can be deceived by the presence of too many risk factors.

In Table 2 we report the number of basis functions calculated for different values of M in the worst case (when the regression date is just before the maturity date).

We recall that for the R-RNN and signature cases all the observations of the underlying asset are included in the basis functions. Loosely speaking we could say that we use the risk-factor set R^4 for such cases.

ρ	**2**	**3**	**4**	**5**	**10**	**20**	**30**
1	3	3	3	3	3	3	3
2	6	6	6	6	6	6	6
3	3	6	10	15	55	210	465
4	1275	1275	1275	1275	1275	1275	1275

Table 2: Number of polynomial basis functions of second degree for different values of M in the worst case (when the regression date is just before the maturity date).

Once we have selected a specific set of risk factors we define the basis functions in the following way

$$\varphi_b(S_{T_1}, \dots, S_{T_i}) := f_b(X_1^\rho, \dots, X_{F_i^\rho}^\rho) \tag{21}$$

where f_b is either a polynomial, a random neural network or an element of the truncated signature. The random variables $\{X_1^\rho, \dots, X_{F_i^\rho}^\rho\}$ are defined as

$$X_j^\rho := \frac{Z_j^\rho - \bar{Z}_j^\rho}{\mathrm{Std}\left[\, Z_j^\rho \,\right]}\,, \quad Z_j^\rho := \log R_j^\rho \tag{22}$$

with $1 \le j \le F_i^\rho$, where R_j^ρ is the j-th risk factor of the set R^ρ, which, in turn, depends on the realization of the price process S_t. Moreover, we denote the empirical mean with a bar over a random variable and the empirical standard deviation with the operator $\mathrm{Std}[\,\cdot\,]$. The normalization of the risk factor is introduced to improve the numerical stability of the linear regression performed in the LSMC method. In the following sections we use the term "basis function" also to refer to f_b, and not only to φ_b.

Finally, in order to significantly increase the efficiency of the algorithms and decrease the computational time, we include in the various regressions only paths for which the options are in the money; see Longstaff et al. (2001).

4.3 Configuration of random networks

We continue the presentation of the algorithms by discussing the architectures of the randomized neural networks, in particular the type of activation function and the distribution from which the parameters are sampled. The number of hidden layers and nodes per layers for all the algorithms will be discussed later on.

We use the leaky ReLU activation function for the R-FFNN and the tanh for the R-RNN; see, also, the choice in Herrera et al. (2024).

The parameters (A, b) of the R-FFNN are sampled using a standard normal distribution with mean 0 and standard deviation 1, which is the standard choice. In general, A and b can be sampled from different distributions that are continuous and have support $\mathbb{R}$. Different hyper-parameters were tested but they did not have a big influence on the results, so we kept the standard choice.

For the R-RNN we use a standard deviation of 0.0001 for A_x and 0.3 for A_h. Also here different hyper-parameters were tested, and the best performing were chosen and used to present the results. In particular, these values are the values in output of a cross-validation procedure, in the sense that we test also different hyper-parameters and the best performing were chosen and used to present the results.

2	**3**	**4**	**5**
12	39	120	363

Table 3: Number of signature basis functions for different values of n, where n denotes the order of the truncated signature, for path of dimension $d = 3$. The results are independent of the windows length and of the risk factor set. The formula for the number of signature basis functions is given by Equation (28).

The parameters θ_i in R-FFNN and R-RNN are determined using 20% of the sampled paths (training data). Given θ_i, the remaining 80% of the sampled paths (evaluation data) are used to compute the option price. Indeed, if the data set was not split, it might happen that the continuation value C_{T_i} depends on future value of the underlying asset, and the neural network can suffer from over-fitting to the training data, by memorizing the paths, instead of learning the continuation value. More precisely, on the training data the approximation of the continuation value C_{T_i} can depend on the future values of paths, since it is trained with them and therefore might remember them. However, by splitting the data into the training and the independent evaluation set, C_{T_i} evaluated on the evaluation set is independent of the future values of paths of the evaluation set. Since the price is only computed on the evaluation set, this effectively prevents this "look into the future" when computing the price. It is, however, possible that methods over-fit on the training data leading to worse results on the evaluation set.

4.4 Configuration of signature methods

For the signature method, we use a truncated signature of order two, three and five. Indeed, the components of the signatures have factorial decay (see Appendix B, Proposition B.3), which means that it is common to choose only the first terms since these will typically be the largest. However, whilst such a truncation captures the largest components, it nevertheless loses information captured by the higher order terms.

For this reason, as explained in Subsection 3.3, we use a suitable augmentation of the dataset before taking the truncated signature. More precisely, we first perform the time-augmentation (Hambly et al. (2010)) by adding time as one of the variables, which ensures that the resulting path is uniquely determined by its signatures. Then, we apply the lead-lag transformation followed by the time-joined transformation in order to embed the time-augmented time series into a continuous path (see Subsection 3.3). In this case, the resulting time series in input of the truncated signature is a $d = 3$ dimensional time time series, whereas in output we have a vector of dimension $s_d(n) = (d^{n+1} - 1)/(d - 1)$, opportunely augmented depending on the payoff under consideration. In Subsection 5.1.4, we will comment on the advantages (or not) of the employed techniques to augment the dataset.

In Table 3 we report the number of basis functions calculated for different values of n, the degree of the truncated signature. Although we show its basis dimensionality, note that we did not employ degree $n = 4$ in our numerical experiments. We show results for dimension $d = 3$, because of the application of the time-augmentation and lead-lag transformation embedding techniques.

As regards the randomized signatures, the different parameters considered in the pricing procedure are the following:

1. ς, which is the variance of the independent normal random variables used to populate the random matrices A_i and biases b_i of Equation (13) used in the algorithm;

2. $k^\star$, which is the dimension of reservoir system, i.e the same matrices and biases found in Equation (13);

3. 'normalization' (shortened to 'norm' in the tables), which defines whether the matrices are normalized, i.e. every entry of the random matrices A_i in Equation (13) is divided by the Frobenius norm of the matrix itself.

5 Numerical investigations

This section reports and discusses the results of the numerical experiments. We investigate the performance of the LSMC method for different choices of basis functions when evaluating prices and sensitivities of Asian and look-back payoffs, the payoffs we introduced in Sections 2.1. We add after this section a further numerical section with a complementary discussion on callable certificates to understand how our results can be extended in a more challenging setting.

The evaluation of all the algorithms is performed on the same computer, equipped with an Apple M1 Pro processor and 16 Gigabyte of RAM. Statistical uncertainties are reported at one-sigma confidence level. Computational times do not include the time for generating the stock paths (around 9 seconds per 10^6 paths), since it is the same for all methods. Indeed, the main interest is in the actual time needed by the algorithms to compute prices from different bases with linear regression, while paths can be generated offline and stored. Numerical procedures, unless specified otherwise, have been implemented in Python.

5.1 Asian options

We go on with discusses the pricing of Asian options with early termination features with both floating and fixed-strike payoff (see Equation (3)), as well as numerical results for sensitivity computation only for the fixed-strike versions of the payoff since the floating-strike ones have trivial Delta and Gamma.

In particular, Table 4 (resp. Table 6) lists option prices obtained for different lengths M of the moving window, obtained via the LSMC method with different choices of basis functions, namely polynomials, R-FFNN, R-RRN, and signatures methods, for Asian options with floating-strike (resp. fixed-strike) payoff. Table 17 (resp. Table 18) and Table 5 (resp. Table 5) report the corresponding uncertainties of Monte Carlo simulations and computational times, respectively.

In all the computations the option maturity is $T = 0.2$ and the number of time steps is $N = 50$. The LSMC results are obtained by batches of simulations with overall $8 \cdot 10^5$ paths to train the regression and $3.2 \cdot 10^6$ to calculate the price.

We continue the analysis by discussing the benchmark models, then we go on with the LSMC algorithm with polynomials, random networks and signatures as basis functions. The numerical results (prices and computational times) can be found for floating-strike options in Tables 4 and 5, while for fixed-strike options in Tables 6 and 7. In Appendix E we display also the corresponding statistical uncertainties in Tables 17 and 18.

5.1.1 Benchmarks

We search the literature for benchmark models. We found three papers focusing on fixed-strike Asian options with observations performed in a moving window: Bernhart et al. (2011), Lelong (2019) and Goudenège et al. (2022). In the first five rows in Tables 4, 5 and 17 we list their results.

Floating-strike Asian option: prices							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
GPR, $P = 250, Q = 8$	1.812	2.676	3.185	3.531	4.319	4.474	4.142
GPR, $P = 1000, Q = 16$	1.873	2.683	3.185	3.535	4.325	4.469	4.141
Binomial Chain	0.940	1.868	2.752	3.323	4.278	4.488	4.163
Bernhart et al. (2011)	1.890	2.684	3.183	3.773	4.268		
Lapeyre et al. (2019)				3.531	4.302		
Polynomial, $\rho = 1, \deg = 2$	1.889	2.682	3.179	3.515	4.225	4.297	3.962
Polynomial, $\rho = 2, \deg = 2$	1.888	2.685	3.191	3.540	4.331	4.468	4.140
Polynomial, $\rho = 3, \deg = 2$	1.889	2.685	3.191	3.540	4.329	4.453	4.108
Polynomial, $\rho = 4, \deg = 2$	1.885	2.679	3.179	3.526	4.295	4.412	4.074
R-FFNN, $\rho = 1, h = 10$	1.889	2.682	3.178	3.515	4.225	4.300	3.965
R-FFNN, $\rho = 2, h = 10$	1.888	2.685	3.190	3.538	4.316	4.448	4.128
R-FFNN, $\rho = 3, h = 10$	1.889	2.685	3.189	3.535	4.247	4.147	3.683
R-FFNN, $\rho = 4, h = 10$	1.888	2.680	3.177	3.514	4.215	4.110	3.666
R-FFNN, $\rho = 1, h = 40$	1.889	2.681	3.177	3.514	4.224	4.301	3.963
R-FFNN, $\rho = 2, h = 40$	1.888	2.684	3.190	3.538	4.327	4.468	4.143
R-FFNN, $\rho = 3, h = 40$	1.888	2.684	3.190	3.538	4.310	4.311	3.715
R-FFNN, $\rho = 4, h = 40$	1.888	2.682	3.180	3.521	4.240	4.146	3.678
R-FFNN, $\rho = 1, h = 120$	1.887	2.680	3.176	3.512	4.223	4.299	3.961
R-FFNN, $\rho = 2, h = 120$	1.887	2.682	3.188	3.535	4.325	4.465	4.136
R-FFNN, $\rho = 3, h = 120$	1.887	2.682	3.188	3.537	4.315	4.401	4.003
R-FFNN, $\rho = 4, h = 120$	1.887	2.682	3.182	3.525	4.252	4.217	3.854
R-FFNN, $\rho = 1, h = 250$	1.886	2.679	3.173	3.510	4.220	4.293	3.959
R-FFNN, $\rho = 2, h = 250$	1.884	2.679	3.183	3.535	4.325	4.465	4.121
R-FFNN, $\rho = 3, h = 250$	1.886	2.679	3.184	3.537	4.315	4.401	4.076
R-FFNN, $\rho = 4, h = 250$	1.886	2.680	3.180	3.525	4.252	4.217	4.002
R-RNN, $\rho = 4, h = 10$	1.885	2.673	3.171	3.507	4.224	4.295	3.935
R-RNN, $\rho = 4, h = 40$	1.883	2.676	3.169	3.505	4.230	4.327	4.005
R-RNN, $\rho = 4, h = 120$	1.881	2.671	3.165	3.502	4.230	4.340	4.029
R-RNN, $\rho = 4, h = 150$	1.883	2.673	3.167	3.502	4.225	4.321	3.992
Signature, $\rho = 4, n = 2$	1.869	2.652	3.159	3.494	4.162	4.105	3.686
Signature, $\rho = 4, n = 3$	1.888	2.682	3.178	3.517	4.232	4.312	4.088
Signature, $\rho = 4, n = 5$	1.885	2.679	3.178	3.522	4.293	4.429	4.134
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=True	1.896	2.694	3.195	3.537	4.277	4.345	4.019
RandSig, $\varsigma = 0.05, k^\star = 20$, norm=True	1.896	2.694	3.196	3.539	4.289	4.376	4.055
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=True	1.896	2.694	3.196	3.542	4.302	4.410	4.064
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=False	1.896	2.694	3.194	3.536	4.267	4.306	3.957
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=False	1.889	2.686	3.188	3.532	4.275	4.384	4.092
RandSig, $\varsigma = 0.3, k^\star = 10$, norm=False	1.893	2.691	3.191	3.533	4.266	4.336	4.015

Table 4: Prices of floating-strike American-style Asian options with maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). Benchmark models: GPR-GHQ and binomial Markov chain of Goudenège et al. (2022), Bernhart et al. (2011), Lelong (2019). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature and randomized signature (RandSig) based methods.

Floating-strike Asian option: computational times							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
GPR, $P = 250, Q = 8$	0	13	11	10	10	10	7
GPR, $P = 1000, Q = 16$	0	320	291	207	203	153	135
Binomial Chain	0	0	0	0	0	0	750
Bernhart et al. (2011)							
Lapeyre et al. (2019)							
Polynomial, $\rho = 1, \deg = 2$	6	6	6	6	6	4	4
Polynomial, $\rho = 2, \deg = 2$	6	8	8	8	7	5	4
Polynomial, $\rho = 3, \deg = 2$	6	52	70	89	179	334	548
Polynomial, $\rho = 4, \deg = 2$	1875	3289	3226	3011	2550	3348	3289
R-FFNN, $\rho = 1, h = 10$	9	10	9	9	9	7	5
R-FFNN, $\rho = 2, h = 10$	9	10	10	10	9	8	6
R-FFNN, $\rho = 3, h = 10$	9	57	76	90	161	254	273
R-FFNN, $\rho = 4, h = 10$	549	554	569	576	593	485	360
R-FFNN, $\rho = 1, h = 40$	18	18	18	18	16	14	9
R-FFNN, $\rho = 2, h = 40$	17	22	19	19	17	13	10
R-FFNN, $\rho = 3, h = 40$	22	66	84	102	170	264	284
R-FFNN, $\rho = 4, h = 40$	588	584	566	567	552	506	346
R-FFNN, $\rho = 1, h = 120$	64	36	72	57	36	25	27
R-FFNN, $\rho = 2, h = 120$	63	40	64	65	34	26	28
R-FFNN, $\rho = 3, h = 120$	61	91	131	145	193	267	287
R-FFNN, $\rho = 4, h = 120$	623	617	704	577	567	490	395
R-FFNN, $\rho = 1, h = 250$	154	149	151	143	132	101	66
R-FFNN, $\rho = 2, h = 250$	143	158	166	65	140	121	73
R-FFNN, $\rho = 3, h = 250$	137	217	213	232	302	352	332
R-FFNN, $\rho = 4, h = 250$	769	901	890	704	677	582	425
R-RNN, $\rho = 4, h = 10$	24	24	24	23	20	17	13
R-RNN, $\rho = 4, h = 40$	50	51	52	53	45	36	28
R-RNN, $\rho = 4, h = 120$	183	170	162	156	151	119	93
R-RNN, $\rho = 4, h = 150$	203	211	204	191	175	145	104
Signature, $\rho = 4, n = 2$	1861	1880	1897	1787	1811	1584	1228
Signature, $\rho = 4, n = 3$	2089	2042	1890	2044	2170	1790	1274
Signature, $\rho = 4, n = 5$	4092	4641	4129	4028	3834	3191	2446
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=True	64	62	61	62	59	55	52
RandSig, $\varsigma = 0.05, k^\star = 20$, norm=True	61	62	62	61	57	53	50
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=True	176	173	181	171	172	142	137
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=False	160	156	157	162	159	144	142
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=False	60	58	58	60	58	52	49
RandSig, $\varsigma = 0.3, k^\star = 10$, norm=False	32	31	30	31	29	27	26

Table 5: Algorithm computational time (in seconds) to price floating-strike American-style Asian options with maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). Benchmark models: GPR-GHQ and binomial Markov chain of Goudenège et al. (2022), Bernhart et al. (2011), Lelong (2019). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature and randomized signature (RandSig) based methods.

Fixed-strike Asian option: prices							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	5.314	5.481	5.559	5.573	5.365	4.545	3.740
Polynomial, $\rho = 2, \deg = 2$	5.313	5.492	5.580	5.624	5.548	4.935	4.186
Polynomial, $\rho = 3, \deg = 2$	5.314	5.492	5.579	5.624	5.545	4.935	4.197
Polynomial, $\rho = 4, \deg = 2$	5.232	5.417	5.514	5.565	5.511	4.926	4.195
R-FFNN, $\rho = 1, h = 10$	5.334	5.508	5.586	5.604	5.380	4.539	3.739
R-FFNN, $\rho = 2, h = 10$	5.330	5.512	5.601	5.642	5.554	4.928	4.182
R-FFNN, $\rho = 3, h = 10$	5.334	5.512	5.595	5.623	5.358	4.783	4.106
R-FFNN, $\rho = 4, h = 10$	3.664	3.861	4.082	4.307	4.920	4.699	4.057
R-FFNN, $\rho = 1, h = 40$	5.329	5.504	5.582	5.601	5.378	4.539	3.739
R-FFNN, $\rho = 2, h = 40$	5.326	5.510	5.599	5.642	5.561	4.944	4.194
R-FFNN, $\rho = 3, h = 40$	5.329	5.509	5.603	5.644	5.547	4.898	4.151
R-FFNN, $\rho = 4, h = 40$	4.197	4.436	4.623	4.818	5.116	4.716	4.054
R-FFNN, $\rho = 1, h = 120$	5.312	5.492	5.570	5.589	5.371	4.537	3.738
R-FFNN, $\rho = 2, h = 120$	5.298	5.489	5.576	5.624	5.553	4.942	4.194
R-FFNN, $\rho = 3, h = 120$	5.312	5.487	5.583	5.631	5.553	4.925	4.187
R-FFNN, $\rho = 4, h = 120$	4.954	5.237	5.399	5.478	5.471	4.884	4.170
R-FFNN, $\rho = 1, h = 250$	5.284	5.467	5.550	5.571	5.358	4.532	3.737
R-FFNN, $\rho = 2, h = 250$	5.268	5.453	5.550	5.596	5.534	4.933	4.192
R-FFNN, $\rho = 3, h = 250$	5.284	5.456	5.553	5.602	5.544	4.931	4.191
R-FFNN, $\rho = 4, h = 250$	5.166	5.389	5.501	5.556	5.504	4.909	4.185
R-RNN, $\rho = 4, h = 10$	5.354	5.524	5.607	5.625	5.421	4.583	3.767
R-RNN, $\rho = 4, h = 40$	5.349	5.524	5.600	5.615	5.393	4.553	3.749
R-RNN, $\rho = 4, h = 120$	5.340	5.514	5.596	5.613	5.392	4.554	3.749
R-RNN, $\rho = 4, h = 150$	5.340	5.518	5.595	5.614	5.389	4.552	3.749
Signature, $\rho = 4, n = 2$	5.156	5.246	5.395	5.458	5.335	4.532	3.823
Signature, $\rho = 4, n = 3$	5.320	5.483	5.560	5.576	5.403	4.785	4.149
Signature, $\rho = 4, n = 3$, no lead-lag	5.266	5.471	5.543	5.573	5.381	4.776	4.146
Signature, $\rho = 4, n = 5$	5.288	5.465	5.561	5.583	5.518	4.899	4.185
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=True	5.319	5.489	5.561	5.579	5.387	4.723	4.046
RandSig, $\varsigma = 0.05, k^\star = 20$, norm=True	5.318	5.489	5.561	5.579	5.388	4.732	4.063
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=True	5.315	5.485	5.558	5.580	5.422	4.824	4.150
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=False	5.286	5.461	5.538	5.560	5.373	4.687	3.970
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=False	4.574	4.510	4.437	4.357	3.934	3.311	3.022
RandSig, $\varsigma = 0.3, k^\star = 10$, norm=False	4.728	4.749	4.746	4.723	4.469	3.829	3.297

Table 6: Prices of fixed-strike American-style Asian options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature and randomized signature (RandSig) based methods.

Fixed-strike Asian option: computational times							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	6	8	6	6	7	4	4
Polynomial, $\rho = 2, \deg = 2$	6	7	7	7	7	6	4
Polynomial, $\rho = 3, \deg = 2$	5	51	71	15	55	210	465
Polynomial, $\rho = 4, \deg = 2$	4719	1275	1275	87	194	368	674
R-FFNN, $\rho = 1, h = 10$	8	10	9	9	10	8	5
R-FFNN, $\rho = 2, h = 10$	9	9	10	9	9	7	6
R-FFNN, $\rho = 3, h = 10$	9	55	73	90	164	255	265
R-FFNN, $\rho = 4, h = 10$	576	566	559	575	536	492	372
R-FFNN, $\rho = 1, h = 40$	31	15	14	15	13	10	8
R-FFNN, $\rho = 2, h = 40$	17	16	17	16	15	11	8
R-FFNN, $\rho = 3, h = 40$	15	56	74	86	145	221	229
R-FFNN, $\rho = 4, h = 40$	505	508	498	492	473	403	302
R-FFNN, $\rho = 1, h = 120$	60	55	55	54	52	37	25
R-FFNN, $\rho = 2, h = 120$	57	55	55	54	53	39	26
R-FFNN, $\rho = 3, h = 120$	55	100	114	130	186	261	245
R-FFNN, $\rho = 4, h = 120$	552	545	547	544	518	428	321
R-FFNN, $\rho = 1, h = 250$	126	114	113	114	101	72	51
R-FFNN, $\rho = 2, h = 250$	126	128	117	122	106	77	54
R-FFNN, $\rho = 3, h = 250$	125	167	180	190	244	301	280
R-FFNN, $\rho = 4, h = 250$	619	604	616	603	555	484	371
R-RNN, $\rho = 4, h = 10$	26	25	30	28	25	21	17
R-RNN, $\rho = 4, h = 40$	60	52	54	53	50	40	34
R-RNN, $\rho = 4, h = 120$	160	163	152	154	141	103	81
R-RNN, $\rho = 4, h = 150$	213	197	211	191	207	163	106
Signature, $\rho = 4, n = 2$	1955	1901	1887	1884	1856	1527	1111
Signature, $\rho = 4, n = 3$	2068	2057	1937	1941	1920	1587	1281
Signature, $\rho = 4, n = 3$, no lead-lag	437	440	420	411	362	278	190
Signature, $\rho = 4, n = 5$	4160	3897	3863	4043	3732	3290	2503
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=True	70	66	64	60	62	55	50
RandSig, $\varsigma = 0.05, k^\star = 20$, norm=True	67	61	59	60	56	54	50
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=True	176	170	168	165	160	150	142
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=False	169	170	168	164	159	146	143
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=False	63	58	58	59	56	52	51
RandSig, $\varsigma = 0.3, k^\star = 10$, norm=False	31	31	30	30	30	27	26

Table 7: Algorithm computational time (in seconds) to price fixed-strike American-style Asian options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature and randomized signature (RandSig) based methods.

In Goudenège et al. (2022) the authors present an efficient method for pricing American-style moving average options based on Gaussian Process Regression and Gauss-Hermite quadrature, thus named GPR-GHQ method (GPR in our tables, for short); P denotes the number of discrete paths for the price process to compute the GPR, whereas Q the number of points employed in the GHQ. Moreover, by exploiting the positive homogeneity of the continuation value, they develop a binomial Markov chain to deal efficiently with medium-long windows (Binomial Chain in our tables). We review both methods in Appendix A. In their numerical investigation, they compare their two methodologies primarily with LSMC. We refer the readers to the authors' original papers for a discussion of the other two contributions.

In detail, among the benchmark models we present GPR-GPQ by Goudenège et al. (2022) with two different choices for parameters P and Q, the method of Bernhart et al. (2011) and the one of Lelong (2019). In our analysis we compare the LSMC method with the benchmark models and we discuss the impact of selecting different sets of basis functions.

5.1.2 Polynomials

The simplest choice of basis functions are polynomials. Longstaff et al. (2001) proposed to use the first three weighted Laguerre polynomials, the first three Hermite polynomials, three trigonometric functions, and simple powers of the underline as basis function obtaining results that are virtually identical to each others. In particular, classical polynomial basis functions up to the second order are the the easiest way to include coupling terms in the basis. The results obtained with the classical polynomials up to degree two were better than with the weighted Laguerre polynomials, therefore we only present these results in our Tables 4 and 6 (in the Tables *deg* refers to the maximum degree of the polynomial). Considerations on the results are consistent for the two types of payoffs.

Empirically, the risk set R^2 is preferred to the risk set R^3, although the latter should contain the best set of information to compute the option prices. In particular, it seems that the choice of R^3 loses effectiveness in case of large values of M. In the case of floating-strike Asian options, the risk set R^2 can beat in some cases also the results obtained with our benchmark model GPR-GPQ. We notice also, by looking at Table 5, that the LSMC with risk set R^2 is to be preferred also in term of calculation speed, especially when compared with the GPR-GPQ method with $P = 1000$ and $Q = 16$. We notice that Goudenège et al. (2022) employ R^3 in their work, thus leading to the wrong conclusion that the continuation value provided by GPR-GPQ is more accurate than that provided by the LSMC method.

In the R^4 case the ability to approximate the continuation value, especially for large values of M is limited by the use of a polynomial of degree two, a limitation imposed by the large size of the problem (1275 in this case).

5.1.3 Randomized neural networks

The choice of polynomials as basis functions may be too simple to catch the non-linear behavior of the continuation value. Thus, we investigate the possibility to replace them by means of a neural network.

We start by discussing R-FFNN basis functions. In order to compare them with polynomials, for a fixed set of risk factors (or features if we wish to use a machine learning terminology), we only change the number of nodes per layer. Indeed, similarly to what found in Herrera et al. (2024), we observed that one hidden layer was sufficient to have a good accuracy (an increase of the number of the hidden layers did not leas to better accuracy). Tables 4 and 6 reports the results.

Quite remarkably, also for the R-FFNN, the results obtained by means of R^2 are slightly higher than those obtained through other risk-factors sets, which indicates that R^2 is more effective than the other bases; this fact is confirmed for every employed value of the hidden size. By comparing the results with polynomial basis functions, we can see that a hidden size equals to 40 is necessary in order to reach the same pricing accuracy. For such value of the hidden size, the computational times of the two pricing techniques are comparable.

Tables 4 and 6 display the results for R-RNNs. We formulate the following observations. First, the hidden size does not seem to affect the results. Second, the computational times grow linearly with the hidden size, and there is a tangible reduction in time with respect to the others methods, when applied to risk set R^4. However, the results obtained with the R-FFNN and polynomials with risk-factor set R^2 are higher than those obtained with R-RNNs.

5.1.4 Signature and randomized signature methods

Tables 4 and 6 display the results. For signatures methods there are relevant gains in the effectiveness if we truncate the signature at order five instead of order two or three. However, similarly to what happens with the R-RNNs, the results obtained with the R-FFNN and polynomials with risk-factor set R^2 are higher.

Before proceeding, the following remark is in order. As said in Section 3.3, one may wonder if the lead-lag transformation is strictly necessary since it duplicates the (spatial) dimension of the time-series. We thus compute prices of fixed-strike American-style options in the same setting of Table 4, with $n = 3$ and without the lead-lag transformation. As you can see the prices obtained with the lead-lag transformation are statistically higher than the ones without such embedding; however, the computational time is much higher.

In the same Tables 4 and 6 we also include randomized signatures, for which a light introduction is provided in Section 3.3.2. We tried this algorithm only for floating-strike and fixed-strike American-style Asian options.

Results show a promising performance for floating-strike options for short-time windows, with respect to standard signature methods, where one can observe that the advantage of using this technique deteriorates while increasing the window length. For the chosen parameters, its prominence decreases for window lengths higher than 5. Results are less clear, but quite similar, for fixed-strike options, although the difference with the polynomial and other methods is less evident. In both cases, it is possible to see that normalization can help the stability of the method and improves pricing results. Computational times are quite high if compared with polynomials on R^2 or low-dimensional neural networks, but much lower when compared with standard signatures. For this reason and the higher calculated derivative prices, randomized signatures are preferred over standard signatures.

5.1.5 Sensitivity computation

The discussion in Section 3.4 show us that finite-difference and Chebyshev methods can be effectively used to calculate sensitivities, although the former method suffers biases, while the regression method seems less reliable. In this section we repeat the analysis for fixed-strike Asian options, by limiting ourself to the first two methods. In Table 8 we can see the results, which are qualitatively similar to the ones of the American put case. In Appendix D we extend the analysis.

The previous results lead us to confirm the use of the Chebyshev method for sensitivity computations. Figure 1 shows the results for Delta and Gamma sensitivities. We calculate option sensitivities for different lengths M of the moving-window by means of the LSMC method with polynomials basis functions with risk factor set R^3. In all the computations the option maturity

Fixed-Strike Asian option: Gamma								
	$S_0 = 85$				$S_0 = 115$			
	Finite Difference		**Chebyshev**		**Finite Difference**		**Chebyshev**	
MC Path	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**
$1.5 \cdot 10^3$	0.0523	0.0144	0.0246	0.0195	0.0189	0.0131	0.0183	0.0123
$7.5 \cdot 10^3$	0.0101	0.0152	0.0108	0.0219	0.0142	0.0129	0.0131	0.0133
$1.5 \cdot 10^4$	0.0145	0.0141	0.0166	0.0179	0.0121	0.0129	0.0153	0.0126
$7.5 \cdot 10^4$	0.0197	0.0189	0.0159	0.0157	0.0150	0.0138	0.0144	0.0121
$1.5 \cdot 10^5$	0.0179	0.0190	0.0165	0.0165	0.0143	0.0153	0.0144	0.0140
$7.5 \cdot 10^5$	0.0163	0.0159	0.0166	0.0168	0.0144	0.0147	0.0144	0.0146
$1.5 \cdot 10^6$	0.0172	0.0174	0.0161	0.0161	0.0145	0.0145	0.0144	0.0147

Table 8: Gamma for a fixed-strike Asian option with moving window $M = 2$, maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. The first column is the total number of simulation paths used by each algorithm. Then, there are two groups each of four columns for different spot prices S_0. In each group the first two columns is the result of the finite-difference method ($\epsilon = 1/8$) with polynomials or R-FFNN basis functions, the last two columns of the Chebyshev method ($\epsilon = 5/8$).

is $T = 0.2$, number of time steps is $N = 50$, the number of paths is $2.5 \cdot 10^5$ both to train the regression and to calculate the price. The sensitivities are calculated by means of the Chebyshev interpolation with an adaptive interpolation interval of ten percent of the spot price according to the algorithm described in Maran et al. (2022).

We can see that the Gamma function is not discontinuous, as in the American put case described in Section 3.4, since in the present case we wait for $M - 1$ days to start the early-termination period, as described in Section 2.1. However, its value abruptly increases spot price around 80% to decrease back to zero at higher spot prices. We notice minor noisy area in calculating the Gamma sensitivity, while Delta is always very smooth.

5.2 Look-back options

We repeat the analysis done in the previous Subsections 5.1 on Asian payoffs in the case of floating-strike and fixed-strike look-back ones. We notice that in this case we replace the arithmetic moving average computed over the interval $[T_{i-M+1}, T_i]$ with the re-scaled minimum or maxima of the asset price when computing the risk factor R^2.

Here, we do not have a comparison with benchmark models in the literature, but we can still compare the results of our LSMC approach based on different choices of the basis functions.

Tables 9 and 11 display the results, while uncertainties of Monte Carlo simulations are reported in Appendix E in Tables 19 and 20. In Tables 10 and 12 are listed the computational times.

First, we can observe that, as for the Asian options, the empirical qualitative results for the fixed-strike payoff and for the floating-strike payoff are very similar. However, in this case, for polynomials results obtained with the risk factor set R^3 are higher than those of R^2, especially for large value of M. For R-FFNN, instead, the risk factor set R^2 is preferred to the risk set R^3; again, a hidden size equals to 40 is necessary in order to reach the same pricing accuracy of polynomials. The result obtained with R-RNNs are smaller than those obtained with the R-FFNN

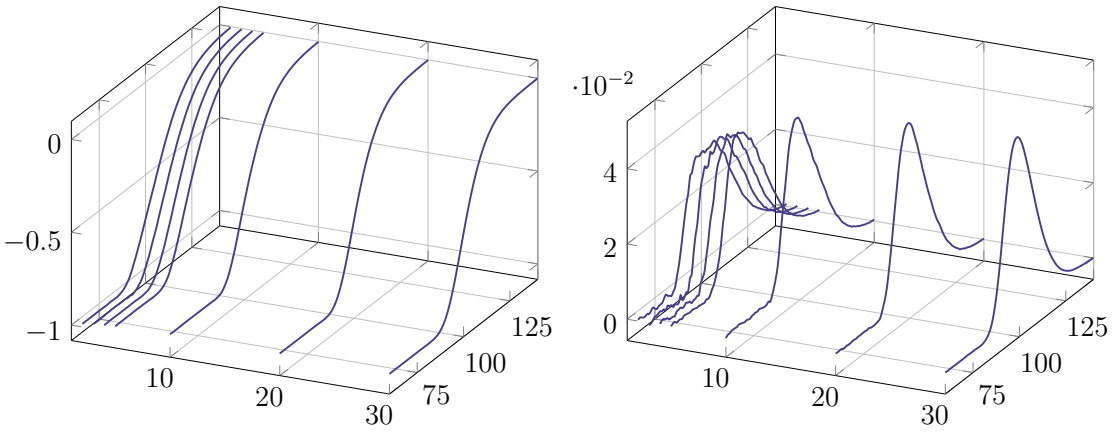

Figure 1: Delta (left panel) and Gamma (right panel) of fixed-strike American-style Asian options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. LSMC model with polynomial base with risk-factor set R^3. On the left axis is reported the window length M, while on the right axis the spot price.

and polynomials. Finally, for the signature methods an order equal to 5 is necessary to achieve the same pricing accuracy of polynomials, at a cost of a very high computational time.

5.2.1 Sensitivity computation

Figure 2 shows the results for Delta and Gamma sensitivities for the fixed-strike version of the payoff. We calculate option sensitivities for different lengths M of the moving-window by means of the LSMC method with polynomials basis functions with risk factor set R^3 with the same settings used for the fixed-strike Asian payoff.

We can see that the Gamma function is continuous, as in the fixed-strike Asian option case, see Section 5.1.5. As in such case we notice minor noisy area in calculating the Gamma sensitivity, while Delta is always very smooth.

Floating-strike look-back option: prices							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	1.889	2.682	3.179	3.515	4.225	4.297	3.962
Polynomial, $\rho = 2, \deg = 2$	1.888	2.685	3.191	3.540	4.331	4.468	4.140
Polynomial, $\rho = 3, \deg = 2$	1.889	2.685	3.191	3.540	4.329	4.453	4.108
Polynomial, $\rho = 4, \deg = 2$	1.885	2.679	3.179	3.526	4.295	4.412	4.074
R-FFNN, $\rho = 1, h = 10$	1.889	2.682	3.178	3.515	4.225	4.299	3.963
R-FFNN, $\rho = 2, h = 10$	1.888	2.684	3.190	3.537	4.309	4.418	4.098
R-FFNN, $\rho = 3, h = 10$	1.889	2.685	3.189	3.536	4.256	4.070	3.637
R-FFNN, $\rho = 4, h = 10$	1.887	2.680	3.176	3.515	4.230	4.174	3.736
R-FFNN, $\rho = 1, h = 40$	1.888	2.681	3.177	3.515	4.225	4.300	3.964
R-FFNN, $\rho = 2, h = 40$	1.888	2.684	3.189	3.537	4.454	4.454	4.132
R-FFNN, $\rho = 3, h = 40$	1.888	2.684	3.189	3.537	4.313	4.389	3.809
R-FFNN, $\rho = 4, h = 40$	1.888	2.682	3.182	3.525	4.252	4.163	3.656
R-FFNN, $\rho = 1, h = 120$	1.887	2.681	3.176	3.512	4.223	4.298	3.961
R-FFNN, $\rho = 2, h = 120$	1.887	2.682	3.187	3.535	4.318	4.456	4.134
R-FFNN, $\rho = 3, h = 120$	1.887	2.682	3.187	3.535	4.312	4.428	4.074
R-FFNN, $\rho = 4, h = 120$	1.887	2.682	3.183	3.528	4.281	4.355	3.996
R-FFNN, $\rho = 1, h = 250$	1.886	2.678	3.173	3.510	4.219	4.294	3.960
R-FFNN, $\rho = 2, h = 250$	1.884	2.679	3.183	3.530	4.309	4.444	4.121
R-FFNN, $\rho = 3, h = 250$	1.886	2.679	3.184	3.531	4.307	4.430	4.093
R-FFNN, $\rho = 4, h = 250$	1.886	2.680	3.182	3.527	4.286	4.394	4.064
R-RNN, $\rho = 4, h = 10$	1.889	2.683	3.182	3.520	4.244	4.269	3.905
R-RNN, $\rho = 4, h = 40$	1.888	2.682	3.177	3.514	4.226	4.297	3.958
R-RNN, $\rho = 4, h = 120$	1.888	2.681	3.176	3.513	4.227	4.297	3.960
R-RNN, $\rho = 4, h = 150$	1.888	2.682	3.176	3.514	4.224	4.298	3.956
Signature, $\rho = 4, n = 2$	1.869	2.652	3.159	3.494	4.162	4.105	3.686
Signature, $\rho = 4, n = 3$	1.888	2.682	3.178	3.517	4.232	4.312	4.088
Signature, $\rho = 4, n = 5$	1.885	2.679	3.176	3.522	4.293	4.429	4.134

Table 9: Prices of floating-strike American-style look-back options with maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

Floating-strike look-back option: computational times							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	5	5	5	5	5	4	3
Polynomial, $\rho = 2, \deg = 2$	6	7	7	7	7	6	4
Polynomial, $\rho = 3, \deg = 2$	5	47	67	81	167	310	493
Polynomial, $\rho = 4, \deg = 2$	4018	4192	3076	5542	6111	5410	4255
R-FFNN, $\rho = 1, h = 10$	8	8	8	8	7	6	5
R-FFNN, $\rho = 2, h = 10$	10	9	8	9	8	6	5
R-FFNN, $\rho = 3, h = 10$	8	49	69	157	150	232	227
R-FFNN, $\rho = 4, h = 10$	529	569	635	646	495	429	319
R-FFNN, $\rho = 1, h = 40$	19	20	17	20	18	14	9
R-FFNN, $\rho = 2, h = 40$	21	22	20	17	16	13	9
R-FFNN, $\rho = 3, h = 40$	16	59	74	95	159	239	238
R-FFNN, $\rho = 4, h = 40$	504	543	533	526	496	432	337
R-FFNN, $\rho = 1, h = 120$	65	61	63	61	57	42	31
R-FFNN, $\rho = 2, h = 120$	71	77	73	71	76	47	34
R-FFNN, $\rho = 3, h = 120$	66	100	118	129	196	262	330
R-FFNN, $\rho = 4, h = 120$	552	539	555	547	540	455	337
R-FFNN, $\rho = 1, h = 250$	147	137	129	132	116	90	57
R-FFNN, $\rho = 2, h = 250$	151	130	120	120	118	81	56
R-FFNN, $\rho = 3, h = 250$	125	171	193	203	252	317	285
R-FFNN, $\rho = 4, h = 250$	631	643	670	647	598	528	398
R-RNN, $\rho = 4, h = 10$	14	14	15	14	14	13	12
R-RNN, $\rho = 4, h = 40$	39	37	36	37	35	28	22
R-RNN, $\rho = 4, h = 120$	136	136	136	129	116	104	78
R-RNN, $\rho = 4, h = 150$	209	186	190	181	177	120	101
Signature, $\rho = 4, n = 2$	1639	1592	1615	1598	4	1288	951
Signature, $\rho = 4, n = 3$	1830	1750	1766	1674	1581	1346	1000
Signature, $\rho = 4, n = 5$	4534	3724	179	3517	3264	2811	2053

Table 10: Algorithm computational time (in seconds) to price floating-strike American-style look-back options with maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

Fixed-strike look-back option: prices							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	4.780	4.684	4.574	4.459	3.827	2.430	1.355
Polynomial, $\rho = 2, \deg = 2$	4.779	4.681	4.574	4.464	3.879	2.633	1.540
Polynomial, $\rho = 3, \deg = 2$	4.780	4.681	4.575	4.467	3.884	2.656	1.571
Polynomial, $\rho = 4, \deg = 2$	4.726	4.630	4.515	4.405	3.844	2.647	1.568
R-FFNN, $\rho = 1, h = 10$	4.796	4.692	4.587	4.474	3.835	2.431	1.358
R-FFNN, $\rho = 2, h = 10$	4.794	4.693	4.587	4.477	3.890	2.638	1.544
R-FFNN, $\rho = 3, h = 10$	4.796	4.693	4.582	4.466	3.714	2.526	1.499
R-FFNN, $\rho = 4, h = 10$	3.725	3.746	3.779	3.773	3.436	2.470	1.480
R-FFNN, $\rho = 1, h = 40$	4.784	4.687	4.583	4.470	3.834	2.429	1.358
R-FFNN, $\rho = 2, h = 40$	4.787	4.686	4.582	4.476	3.895	2.655	1.553
R-FFNN, $\rho = 3, h = 40$	4.784	4.689	4.584	4.478	3.887	2.615	1.519
R-FFNN, $\rho = 4, h = 40$	4.067	3.995	3.995	3.969	3.554	2.489	1.478
R-FFNN, $\rho = 1, h = 120$	4.768	4.671	4.568	4.457	3.826	2.425	1.355
R-FFNN, $\rho = 2, h = 120$	4.769	4.672	4.565	4.459	3.888	2.654	1.552
R-FFNN, $\rho = 3, h = 120$	4.768	4.672	4.567	4.464	3.887	2.647	1.554
R-FFNN, $\rho = 4, h = 120$	4.564	4.496	4.413	4.325	3.791	2.577	1.522
R-FFNN, $\rho = 1, h = 250$	4.752	4.653	4.550	4.440	3.814	2.419	1.351
R-FFNN, $\rho = 2, h = 250$	4.745	4.651	4.541	4.437	3.869	2.647	1.550
R-FFNN, $\rho = 3, h = 250$	4.751	4.647	4.544	4.439	3.874	2.653	1.568
R-FFNN, $\rho = 4, h = 250$	4.751	4.678	4.589	4.490	3.814	2.597	1.541
R-RNN, $\rho = 4, h = 10$	4.813	4.712	4.599	4.488	3.857	2.454	1.372
R-RNN, $\rho = 4, h = 40$	4.806	4.705	4.597	4.486	3.848	2.438	1.366
R-RNN, $\rho = 4, h = 120$	4.793	4.698	4.596	4.485	3.845	2.437	1.365
R-RNN, $\rho = 4, h = 150$	4.788	4.695	4.592	4.478	3.844	2.437	1.364
Signature, $\rho = 4, n = 2$	4.702	4.591	4.432	4.353	3.787	2.419	1.404
Signature, $\rho = 4, n = 3$	4.778	4.679	4.567	4.454	3.803	2.553	1.500
Signature, $\rho = 4, n = 5$	4.754	4.654	4.544	4.430	3.867	2.634	1.563

Table 11: Prices of fixed-strike American-style look-back options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

Fixed-strike look-back option: computational times							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	4	4	4	4	4	3	3
Polynomial, $\rho = 2, \deg = 2$	5	6	6	6	6	5	3
Polynomial, $\rho = 3, \deg = 2$	4	45	62	79	164	303	474
Polynomial, $\rho = 4, \deg = 2$	9360	6870	6076	7441	5328	5463	5315
R-FFNN, $\rho = 1, h = 10$	7	7	6	6	6	5	4
R-FFNN, $\rho = 2, h = 10$	9	8	7	7	7	5	4
R-FFNN, $\rho = 3, h = 10$	7	47	66	78	149	223	229
R-FFNN, $\rho = 4, h = 10$	514	529	515	505	492	422	335
R-FFNN, $\rho = 1, h = 40$	18	17	15	16	13	10	7
R-FFNN, $\rho = 2, h = 40$	17	18	17	16	15	11	7
R-FFNN, $\rho = 3, h = 40$	16	59	74	92	166	229	229
R-FFNN, $\rho = 4, h = 40$	510	524	524	514	493	415	303
R-FFNN, $\rho = 1, h = 120$	54	50	46	44	37	23	15
R-FFNN, $\rho = 2, h = 120$	56	53	50	47	39	26	16
R-FFNN, $\rho = 3, h = 120$	51	93	108	120	174	251	227
R-FFNN, $\rho = 4, h = 120$	553	549	554	539	501	438	338
R-FFNN, $\rho = 1, h = 250$	104	106	91	89	81	49	34
R-FFNN, $\rho = 2, h = 250$	105	103	103	98	76	55	32
R-FFNN, $\rho = 3, h = 250$	116	161	157	168	232	269	258
R-FFNN, $\rho = 4, h = 250$	112	583	583	616	567	467	347
R-RNN, $\rho = 4, h = 10$	19	20	19	18	17	14	11
R-RNN, $\rho = 4, h = 40$	43	39	37	39	31	28	20
R-RNN, $\rho = 4, h = 120$	140	124	123	115	95	69	47
R-RNN, $\rho = 4, h = 150$	176	162	150	141	121	89	63
Signature, $\rho = 4, n = 2$	1656	1843	1791	1569	1506	1247	972
Signature, $\rho = 4, n = 3$	1776	1721	1677	1667	1557	1346	981
Signature, $\rho = 4, n = 5$	3438	3497	3466	3411	3139	2668	1984

Table 12: Algorithm computational time (in seconds) to price fixed-strike American-style look-back options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

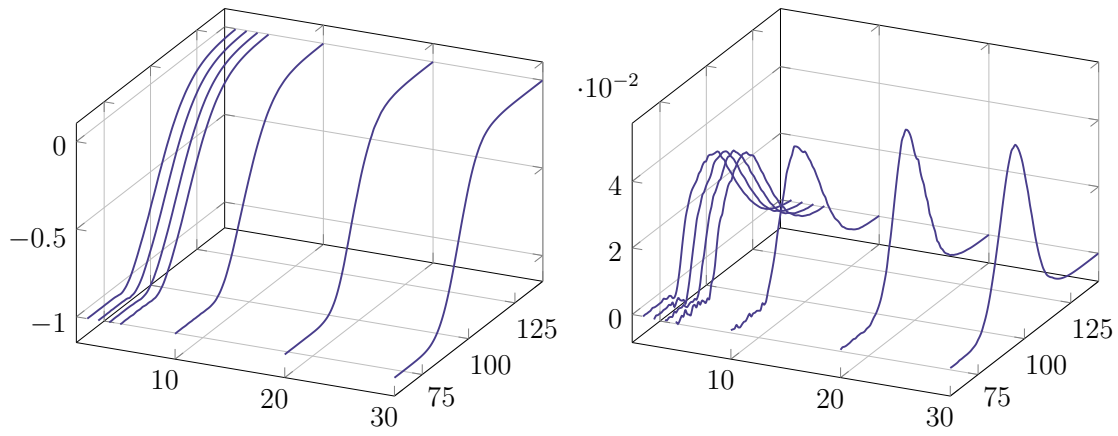

Figure 2: Delta (left panel) and Gamma (right panel) of fixed-strike American-style look-back options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. LSMC model with polynomial base with risk-factor set R^3. On the left axis is reported the window length M, while on the right axis the spot price.

6 Callable certificates

In the previous numerical section we have seen that polynomial basis functions or random networks, in particular the R-FFNN architecture, seem the best choices whenever we can prune the input features of unnecessary information, as we do when we choose the risk factor R^2.

However, we must exploit the specific form tof the payoff to devise the right risk-factor set. In general, if we deal with more complex situations, we could not introduce such simplifications. In order to understand the challenge of this cases we analyze a class of investment products actively traded in the market: callable certificates.

6.1 Product description

Certificates are structured investment products which usually contains optional features automatically triggered based on a specific event. A common type of equity certificate is the auto-callable product. These products are triggered on predetermined observation dates if an underlying asset or reference portfolio reaches or surpasses a barrier. Upon termination, the investor receives the principal investment amount plus a coupon. If the product is not terminated early, the investor receives the principal amount, or a portion thereof, along with an optional payoff on the maturity date. This feature is frequently offered in low-yield markets, providing the investor with the potential for an above-market yield, albeit with the risk of not receiving a coupon. We refer the readers to Deng et al. (2016), Alm et al. (2013), and Farkas et al. (2022) and references therein for a description of this product.

In this study, we analyze equity certificates that feature an early-termination clause instead of an auto-callable feature. This allows the issuer to terminate the product on predetermined observation dates at their discretion, making them "callable certificates." This enables the issuer to have greater control over the marking of their funding benefits, while the investor still has the opportunity to re-enter a high-yield structure on termination dates. There is a limited literature on these new products. We refer the readers to Agrawal et al. (2022) and references therein for further details.

In this section we present the numerical results for "snowball" and "lock-in" callable certifi-

Certificate: prices	**Snowball**			**Lock-in**		
Model	**1y**	**2y**	**5y**	**1y**	**2y**	**5y**
Polynomial, deg = 2	1.00025	1.00006	1.00448	1.00019	0.99998	1.00021
Polynomial, deg = 3	1.00025	0.99921	0.99600	1.00023	0.99897	0.99652
Polynomial, deg = 5	1.00025	1.00017		1.00019	0.99859	
R-FFNN, $h = 10$	1.00025	0.99998	1.00382	1.00019	0.99843	0.99555
R-FFNN, $h = 50$	0.99987	0.99876	0.99934	1.00019	0.99759	0.99240
R-FFNN, $h = 120$	0.99924	0.99896	0.99650	1.00019	0.99743	0.99176
R-RNN, $h = 10$	1.00025	0.99918	1.00409	0.99291	0.99862	0.99547
R-RNN, $h = 50$	1.00025	0.99887	1.00374	0.99302	0.99819	0.99498
R-RNN, $h = 120$	1.00025	0.99888	1.00319	0.99287	0.99825	0.99568
Signature, $n = 2$	1.00022	0.99964	0.99942	1.00019	1.00022	1.00052
Signature, $n = 3$	1.00024	1.00015	1.00060	1.00032	1.00010	0.99821
Signature, $n = 5$	1.00026	0.99987	1.00271	1.00028	1.00016	0.99814

Table 13: Prices of snowball and lock-in certificates with maturity of 1, 2 and 5 years (other characteristics described in the text). Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

cates. The specifics of the payoffs and the corresponding dynamic programming problem are presented in Appendix C. In this numerical section we assume that the underlying asset follows the dynamics described in Section 4.1.

We perform all the numerical investigations by using a LSMC method with different choices of the basis functions. As risk factors we use at regression date T_i, with $1 \leq i \leq N$ and $T_N = T$, a set formed by the observations of the underlying price processes on all the coupon dates up to the regression date.

6.2 Snowball payoff

We consider a snowball payoff with different maturities which pays quarterly a coupon if the underlying asset performance is above 100%. In particular we choose the following characteristics: (i) maturity of one year with coupon equal to 2.3% and capital barrier at 35%, (ii) maturity of two years with coupon equal to 2.4% and capital barrier at 30%, (iii) maturity of five years with coupon equal to 2.85% and capital barrier at 30%. The characteristics of the product are chosen so that the initial price is approximately at par, when evaluated with the LSMC method with polynomial basis on the risk set described at the beginning of Section 6.

In the first three columns of Table 13 we report the prices of the snowball certificates defined above by changing the basis functions used in the LSMC method to price the products. We can see that the prices are closely aligned, and within the Monte Carlo uncertainties, reported in appendix in Table 21. In Table 14 are listed the computational times. We can see the best choice of basis functions is R-FFNN and R-RNN (in particular for longer maturities), followed by polynomials. In particular, the last ones become unfeasible for many dates and larger degrees (in the Tables *deg* refers to the maximum degree of the polynomial).

Then, we investigate also the performance of finite-difference and Chebyshev methods in computing sensitivities. In Table 15 we list the results. We analyze different algorithms, even if we

Certificate: times	**Snowball**			**Lock-in**		
Model	**1y**	**2y**	**5y**	**1y**	**2y**	**5y**
Polynomial, deg = 2	1	3	27	1	3	30
Polynomial, deg = 3	1	5	310	1	6	363
Polynomial, deg = 5	2	54		2	37	
R-FFNN, $h = 10$	1	2	10	1	2	8
R-FFNN, $h = 50$	3	4	18	3	5	17
R-FFNN, $h = 120$	5	10	36	5	12	36
R-RNN, $h = 10$	1	2	8	1	2	8
R-RNN, $h = 50$	3	8	31	3	9	32
R-RNN, $h = 120$	8	23	100	8	27	99
Signature, $n = 2$	30	94	299	44	136	434
Signature, $n = 3$	32	96	309	47	144	461
Signature, $n = 5$	55	189	1066	73	253	1145

Table 14: Algorithm computational time (in seconds) to price snowball and lock-in certificates with maturity of 1, 2 and 5 years (other characteristics described in the text). Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

have seen in Table 13 that prices are not in perfect agreement, to understand the stability of sensitivities. Polynomials basis function are taken up to third order; random neural networks have 120 inner nodes; signature methods are truncated at third degree with lead-lag and time embedding. We can see that, for snowball certificates, both finite-difference and Chebyshev methods seem lead to similar results in term of stability. In Appendix D we extend the analysis.

6.3 Lock-in payoff

We consider a lock-in payoff with different maturities which pays quarterly a coupon if the underlying asset performance is above different coupon barriers. In particular we choose the following characteristics: (i) maturity of one year with coupon equal to 2.8%, coupon barrier at 100% and capital barrier at 40%, (ii) maturity of two years with coupon equal to 2.4%, coupon barrier at 90% and capital barrier at 30%, (iii) maturity of five years with coupon equal to 3%, coupon barrier at 90% and capital barrier at 30%. The characteristics of the product are chosen so that the initial price is approximately at par, when evaluated with the LSMC method with polynomial basis on the risk set described at the beginning of Section 6.

In the last three columns of Table 13 we report the prices of the lock-in certificates defined above by changing the basis functions used in the LSMC method to price the products. We can see that the prices are closely aligned, and within the Monte Carlo uncertainties, reported in appendix in Table 21. In Table 14 are listed the computational times. We can see that the best choices of basis functions are R-FFNN and R-RNN, the former being faster for shorter maturities but less accurate, followed by polynomials, although the last ones become unfeasible for many dates and larger degrees. Here, we consider more accurate a price which is lower than the others, since the dynamic program terminates the options to minimize the price.

Then, we investigate also the performance of finite-difference and Chebyshev methods in computing sensitivities. In Table 16 we list the results. Polynomials basis function are taken up to

Snowball certificate: Gamma								
	$S_0 = 85$				$S_0 = 115$			
	Finite Difference		**Chebyshev**		**Finite Difference**		**Chebyshev**	
MC Path	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**
$1.5 \cdot 10^3$	−0.7805	−0.2597	0.6228	0.5144	−0.2670	−0.3402	−0.3270	−0.5175
$7.5 \cdot 10^3$	−0.1612	−0.2667	−1.0934	−0.5321	−0.4461	−0.4960	−0.4202	−0.4462
$1.5 \cdot 10^4$	−0.6732	−0.5665	−0.2835	−0.3754	−0.5005	−0.4743	−0.3645	−0.3401
$7.5 \cdot 10^4$	−0.7944	−0.7627	−0.7459	−0.7057	−0.3579	−0.3502	−0.4184	−0.3662
$1.5 \cdot 10^5$	−0.6740	−0.6673	−0.8766	−0.8066	−0.3287	−0.3504	−0.3843	−0.3882
$7.5 \cdot 10^5$	−0.7060	−0.6997	−0.5815	−0.5753	−0.3956	−0.3801	−0.3904	−0.3923
$1.5 \cdot 10^6$	−0.6721	−0.6454	−0.6724	−0.6656	−0.3774	−0.3698	−0.3999	−0.3835
MC Path	**RNN**	**Sign.**	**RNN**	**Sign.**	**RNN**	**Sign.**	**RNN**	**Sign.**
$1.5 \cdot 10^3$	0.1463	−0.8456	−0.0644	0.4109	−0.1973	−0.2991	−0.2242	−0.3014
$7.5 \cdot 10^3$	−0.1632	−0.2044	−1.1572	−1.0406	−0.4577	−0.5115	−0.5001	−0.5467
$1.5 \cdot 10^4$	−0.7240	−0.6948	−0.2780	−0.3029	−0.4466	−0.5901	−0.3914	−0.3985
$7.5 \cdot 10^4$	−0.8369	−0.7635	−0.7245	−0.7360	−0.3346	−0.3849	−0.3717	−0.4568
$1.5 \cdot 10^5$	−0.7406	−0.6527	−0.9293	−0.8558	−0.3273	−0.3956	−0.3519	−0.4479
$7.5 \cdot 10^5$	−0.7535	−0.6895	−0.6401	−0.5684	−0.3702	−0.4545	−0.3659	−0.4553
$1.5 \cdot 10^6$	−0.7159	−0.6578	−0.7314	−0.6569	−0.3590	−0.4396	−0.3660	−0.4763

Table 15: Gamma for a snowball certificate with maturity of two years (multiplied by 10^4). The first column is the total number of simulation paths used by each algorithm. Then, there are two groups each of four columns for different spot prices S_0. In each group the first two columns is the result of the finite-difference method ($\epsilon = 1/4$) with different basis functions, the last two columns of the Chebyshev method ($\epsilon = 3/4$).

Lock-in certificate: Gamma								
	$S_0 = 85$				$S_0 = 115$			
	Finite Difference		**Chebyshev**		**Finite Difference**		**Chebyshev**	
MC Path	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**	**Polyn.**	**FFNN**
$1.5 \cdot 10^3$	−1.2744	−0.5865	−0.2483	0.7709	−0.3120	−0.2653	−0.0435	0.4992
$7.5 \cdot 10^3$	−1.0049	−0.7238	−0.5390	−0.1646	−0.3930	−0.3901	−0.2683	−0.3205
$1.5 \cdot 10^4$	−0.5322	−0.3719	−0.8770	−0.5917	−0.4126	−0.4536	−0.2543	−0.3525
$7.5 \cdot 10^4$	−0.6925	−0.6738	−0.6965	−0.6852	−0.3510	−0.3871	−0.2889	−0.3793
$1.5 \cdot 10^5$	−0.7210	−0.6625	−0.7761	−0.7591	−0.3529	−0.3937	−0.3092	−0.3912
$7.5 \cdot 10^5$	−0.6939	−0.6155	−0.6865	−0.5694	−0.3452	−0.3903	−0.2991	−0.3768
MC Path	**RNN.**	**Sign.**	**RNN**	**Sign.**	**RNN**	**Sign.**	**RNN**	**Sign.**
$1.5 \cdot 10^3$	−0.6832	−1.3537	−0.0323	−0.0599	−0.2726	−0.1899	0.2069	−0.0256
$7.5 \cdot 10^3$	−0.7030	−1.0818	−0.0838	−0.6224	−0.3542	−0.2844	−0.2192	−0.1320
$1.5 \cdot 10^4$	−0.5318	−0.5833	−0.3691	−0.8535	−0.4088	−0.3204	−0.2782	−0.1588
$7.5 \cdot 10^4$	−0.7744	−0.7500	−0.8087	−0.6655	−0.3687	−0.2103	−0.3562	−0.1806
$1.5 \cdot 10^5$	−0.7341	−0.7082	−0.8590	−0.7330	−0.3776	−0.2132	−0.3623	−0.1909
$7.5 \cdot 10^5$	−0.6806	−0.6533	−0.6509	−0.5739	−0.3688	−0.2080	−0.3495	−0.1620

Table 16: Gamma for a lock-in certificate with maturity of two years (multiplied by 10^4). The first column is the total number of simulation paths used by each algorithm. Then, there are two groups each of four columns for different spot prices S_0. In each group the first two columns is the result of the finite-difference method ($\epsilon = 1/4$) with different basis functions, the last two columns of the Chebyshev method ($\epsilon = 3/4$).

third order; random neural networks have 120 inner nodes; signature methods are truncated at third degree with lead-lag and time embedding. As in the previous case of snowball certificates, we obtain a similar behavior in term of stability both for finite-difference and Chebyshev algorithms. In Appendix D we extend the analysis.

7 Conclusion and Further Developments

In the present paper, we studied state of the art algorithms, that are now classified under the name of machine learning, to price Asian, look-back products, and callable certificates with early-termination features. More precisely, we adapted the approach of Herrera et al. (2024) to the case of path-dependent payoffs with early-termination features, and we compare the results with alternative original formulations based on signature methods. All the experiments are run under a Black-Scholes dynamics for a single stock price.

We observed that, at least for the type of payoffs and the dynamics we considered, these algorithms have a performance, in terms of accuracy and computational efficiency, often comparable to that of more traditional algorithms, such as the ones based on polynomial basis functions. In particular, we observed that a careful selection of the risk factors in traditional approaches may lead to relevant improvement in computational time by saving the accuracy of the results, at least when pricing Asian and look-back products with early-termination features. On the other

hand, when pricing callable certificates, the best choices of basis functions are random networks (R-FFNN and R-RNN); in this case, polynomials become unfeasible for many dates and larger degrees. In this context, signatures do not emerge as strictly necessary algorithms. Although prices obtained from truncated signatures are among the highest for look-back options and certificates, these are matched by R-FFNNs which, moreover, have much lower computational times. Further, more investigations would be needed to address the computational issues (inversion of the regression basis) which turns out to be (almost) singular[1] with regularization techniques (e.g. Lasso or Ridge regularization). Such problems can be avoided using randomized signatures, as alluded to in Section 3.3.2, for which we can obtain similar, though not equal, results. Standard signatures show their abilities as feature extractors particularly well for high truncations, and that is where advantages compared to randomized signatures become more apparent, but the latter have the advantage of being able to offer similar expressiveness at lower computational costs.

As a second contribution, we analyzed algorithms to compute sensitivities; in particular, we found that Chebyshev interpolation techniques are an effective choice for Delta and Gamma calculations. The strength of the method relies in decoupling the selection of the interpolation interval from the computation of the derivatives, so that a larger size of the interval, required to stabilize the sensitivity computation, does not introduce biases. To the best of our knowledge, it is the first time that first and second order Greeks are computed for Amerasian derivatives and callable certificates.

As further developments, we plan to investigate the performance of machine learning based methods for more complicated, e.g., non-Markov dynamics, and in higher dimensional settings.

Acknowledgements

All the authors warmly thank Antonino Zanette (University of Udine) for having provided the MATLAB code for the implementation of the benchmark methodologies used for Asian payoffs. They thank Qi Feng (Florida State University) for having provided insights on the usefulness or not of the lead-lag transformation. They thank also Francesca Sivero for fruitful discussions on design patterns and Giuseppe Di Poto for American options calculations with binomial trees.

Disclaimer

The authors report no potential competing interests. The opinions expressed in this document are solely those of the authors and do not represent in any way those of their present and past employers.

References

Agrawal, B. and L. D. Li (2022). “Fixed income optimal suboptimality”. In: *Risk Magazine* 11.

Akyildirim, Erdinc, Matteo Gambara, Josef Teichmann, and Syang Zhou (2022). “Applications of Signature Methods to Market Anomaly Detection”. Preprint arXiv:2201.02441.

— (2023). “Randomized Signature Methods in Optimal Portfolio Selection”. Preprint arXiv:2312.16448.

[1] This phenomenon is already known in the literature and is due to the fact that some signature coefficients of embedded paths may be equal (see, for example, Section 2.1.5 in Chevyrev et al. (2016) - version 1).

Alm, T., B. Harrach, D. Harrach, and M. Keller (2013). "A Monte Carlo pricing algorithm for autocallables that allows for stable differentiation". In: *J. Comput. Finance* 17.1, pp. 43–70.

Barraquand, J. and D. Martineau (1995). "Numerical Valutation of High Dimensional Multivariate American Securities". In: *Journal of Financial and Quantitative Analysis* 30, pp. 383–405.

Barrera-Esteve, C., F. Bergeret, C. Dossal, E. Gobet, A. Meziou, R. Munos, and D. Reboul-Salze (2006). "Numerical methods for the pricing of Swing options: a stochastic control approach". In: *Methodology and Computing in Applied Probability* 8.4, pp. 517–540.

Bayraktar, E., Q. Feng, and Z. Zhang (2022). "Deep Signature Algorithm for Path-Dependent American option pricing". Preprint arXiv:2211.11691.

Becker, S., P. Cheridito, and A. Jentzen (2020). "Pricing and Hedging American Style Options with Deep Learning". In: *J. Risk Financial Manag.* 13.7, p. 158.

Bernard, Carole, Anne MacKay, and Max Muehlbeyer (2014). "Optimal surrender policy for variable annuity guarantees". In: *Insurance: Mathematics and Economics* 55, pp. 116–128.

Bernhart, M., P. Tankov, and X. Warin (2011). "A finite-dimensional approximation for pricing moving average options". In: *Sifin* 2.1, pp. 989–1013.

Biagini, Francesca, Lukas Gonon, and Niklas Walter (2024). "Universal randomised signatures for generative time series modelling". Preprint arXiv:2406.10214.

Bilger, R. (2003). "Valuation of American-Asian Options with the Longstaff-Schwartz Algorithm". MSc Thesis, Oxford University.

Black, F. and M. Scholes (1973). "The pricing of options and corporate liabilities". In: *J. Pol. Econ.* 81.3, pp. 637–654.

Boedihardjo, Horatio, Xi Geng, Terry Lyons, and Danyu Yang (2016). "The signature of a rough path: Uniqueness". In: *Advances in Mathematics* 293, pp. 720–737. DOI: 10.1016/j.aim.2016.02.011.

Cao, Weipeng, Xizhao Wang, Zhong Ming, and Jinzhu Gao (2018). "A review on neural networks with random weights". In: *Neurocomputing* 275, pp. 278–287.

Chen, K. T. (1957). "Integration of paths, geometric invariants and a generalized Baker-Hausdorff formula". In: *Annals of Mathematics*, pp. 163–178.

— (1977). "Iterated path integrals". In: *Bulletin of the American Mathematical Society* 83.5, pp. 831–879.

Chevyrev, Ilya and Andrey Kormilitzin (2016). "A Primer on the Signature Method in Machine Learning". Preprint arXiv:1603.03788.

Compagnoni, Enea Monzio, Anna Scampicchio, Luca Biggio, Antonio Orvieto, Thomas Hofmann, and Josef Teichmann (2023). "On the effectiveness of Randomized Signatures as Reservoir for Learning Rough Dynamics". en. In: *2023 International Joint Conference on Neural Networks (IJCNN)*. International Joint Conference on Neural Networks (IJCNN 2023); Conference Location: Gold Coast, Australia; Conference Date: June 18-23, 2023. Piscataway, NJ: Ieee, pp. 1–8. ISBN: 978-1-6654-8867-9. DOI: 10.1109/ijcnn54540.2023.10191624.

Cuchiero, Christa, Lukas Gonon, Lyudmila Grigoryeva, Juan-Pablo Ortega, and Josef Teichmann (2021). "Expressive Power of Randomized Signature". In: *The Symbiosis of Deep Learning and Differential Equations*.

— (2022). "Discrete-Time Signatures and Randomness in Reservoir Computing". In: *IEEE Transactions on Neural Networks and Learning Systems* 33.11, pp. 6321–6330. DOI: 10.1109/tnnls.2021.3076777.

Dai, M., P. Li, and J. E. Zhang (2010). "A lattice algorithm for pricing moving average barrier options". In: *Journal of Economic Dynamics and Control* 34.3, pp. 542–554.

Daluiso, Roberto, Emanuele Nastasi, Andrea Pallavicini, and Giulio Sartorelli (2024). "Swing option pricing consistent with futures smiles". In: *Applied Stochastic Models in Business and Industry* 40.2, pp. 224–242. DOI: 10.1002/asmb.2747.

Deng, G., J. Mallett, and C. McCann (2016). "Modeling autocallable structured products". In: *J. Deriv. Hedge Funds* 17, pp. 323–344.

Dirnstorfer, S., A. J. Grau, and R. Zagst (2013). "High-dimensional regression on sparse grids applied to pricing moving window Asian options". In: *Open Journal of Statistics* 3.6, pp. 427–440.

Farkas, W., F. Ferrari, and U. Ulrych (2022). "Pricing autocallables under local-stochastic volatility". In: *Frontiers of Mathematical Finance* 1.4, pp. 575–610.

Federico, S. and P. Tankov (2015). "Finite-dimensional representations for controlled diffusions with delay". In: *Applied Mathematics & Optimization* 71, pp. 165–194.

Feng, Q., M. Luo, and Z. Zhang (2021). "Deep Signature FBSDE Algorithm". Preprint arXiv:2108.10504.

Fermanian, Adeline (2021). "Embedding and learning with signatures". In: *Computational Statistics & Data Analysis* 157, p. 107148.

Flint, Guy, Ben Hambly, and Terry Lyons (2016). "Discretely sampled signals and the rough Hoff process". In: *Stochastic Processes and their Applications* 126.9, pp. 2593–2614.

Gaß, M., K. Glau, M. Mahlstedt, and M. Mair (2016). "Chebyshev interpolation for parametric option pricing". In: *Finance and Stochastics* 22.3, pp. 701–731.

Goudenège, Ludovic, Andrea Molent, and Antonino Zanette (2022). "Moving average options: Machine learning and Gauss-Hermite quadrature for a double non-Markovian problem". In: *European Journal of Operational Research* 303.2, pp. 958–974.

Grau, A. J. (2008). "Applications of least-squares regressions to pricing and hedging of financial derivatives". PhD Thesis, Technische Universität München.

Gyurkó, L. G., T. Lyons, M. Kontkowski, and J. Field (2013). "Extracting information from the signature of a financial data stream". Preprint arXiv:1307.7244.

Hambly, B. and T. Lyons (2010). "Uniqueness for the signature of a path of bounded variation and the reduced path group". In: *Annals of Mathematics*, pp. 109–167.

Herrera, Calypso, Florian Krach, Pierre Ruyssen, and Josef Teichmann (2024). "Optimal stopping via randomized neural networks". In: *Frontiers of Mathematical Finance* 3.1, pp. 31–77.

Hoff, Benjamin (2006). "The Brownian frame process as a rough path". Preprint arXiv:math/0602008.

Johnson, William B. and Joram Lindenstrauss (1984). "Extensions of Lipschitz mappings into Hilbert space". In: *Contemporary mathematics* 26, pp. 189–206.

Kao, Chih-Hao and Yuh-Dauh Lyuu (2003). "Pricing of moving-average-type options with applications". In: *Journal of Futures Markets* 23.5, pp. 415–440.

Király, Franz J. and Harald Oberhauser (2019). "Kernels for Sequentially Ordered Data". In: *Journal of Machine Learning Research* 20.31, pp. 1–45.

Kohler, M., A. Krzyżak, and N. Z. Todorovic (2010). "Pricing of High-Dimensional American Options by Neural Networks". In: *Math. Finance* 20.3, pp. 383–410.

Lapeyre, B. and J. Lelong (2021). "Neural network regression for Bermudan option pricing". In: *Monte Carlo Methods Appl.* 27.3, pp. 227–247.

Lelong, J. (2019). "Pricing path-dependent Bermudan options using Wiener chaos expansion: an embarrassingly parallel approach". Preprint arXiv:1901.05672.

Létourneau, P. and L. Stentoft (2023). "Simulated Greeks for American options". In: *Quantitative Finance* 23.4, pp. 653–676. DOI: 10.1080/14697688.2022.2159869.

Levin, D., T. Lyons, and H. Ni (2013). "Learning from the past, predicting the statistics for the future, learning an evolving system". Preprint arXiv:1309.0260.

Longstaff, F. A. and E. S. Schwartz (2001). "Valuing American Options by Simulation: A Simple Least-Squares Approach". In: *Rev. Financ. Stud.* 14.1, pp. 113–147.

Lyons, T. J. (1998). "Differential equations driven by rough signals". In: *Revista Matemática Iberoamericana* 14.2, pp. 215–310.

Lyons, Terry (2014). "Rough paths, signatures and the modelling of functions on streams". Preprint arXiv:1405.4537.

Lyons, Terry, Sina Nejad, and Imanol Perez Arribas (2019). "Numerical method for model-free pricing of exotic derivatives in discrete time using rough path signatures". In: *Applied Mathematical Finance* 26.6, pp. 583–597.

Maran, A., A. Pallavicini, and S. Scoleri (May 2022). "Chebyshev Greeks: smoothing gamma without bias". In: *Risk Magazine*.

Moor, Michael, Max Horn, Christian Bock, Karsten Borgwardt, and Bastian Rieck (2020). "Path Imputation Strategies for Signature Models of Irregular Time Series". Preprint arXiv:2005.12359.

Pannier, Alexandre and Cristopher Salvi (2024). "A path-dependent PDE solver based on signature kernels". Preprint arXiv:2403.11738.

Sabate-Vidales, Marc, David Šiška, and Lukasz Szpruch (2020). "Solving path dependent PDEs with LSTM networks and path signatures". Preprint arXiv:2011.10630.

Thompson, A. C. (1995). "Valuation of path-dependent contingent claims with multiple exercise decisions over time: the case of Take or Pay". In: *Journal of Financial and Quantitative Analysis* 30, pp. 271–293.

Tilley, J. (1993). "Valuing American Options in a Path Simulation Model". In: *Transactions of the Society of Actuaries* 45, pp. 85–104.

A The GPR-GHQ algorithm and the binomial Markov chain method

We discuss both the GPR-GHQ algorithm and the binomial Markov chain method presented in Goudenège et al. (2022). We start with the description of the GPR-GHQ algorithm. We remind that GPR stands for Gaussian Process Regression and GHQ for Gauss-Hermite quadrature. It is a backward induction algorithm that employs the GHQ to compute the continuation value of the option only for some particular path of the underlying and the GPR to extrapolate the whole continuation value from those observations.

A.1 Details on the GPR-GHQ algorithm

Precisely, let N be the number of time steps and T_i as in Subsection 2.1. In addition, let $(\mathbf{A}_i)_{M\leq i\leq N}$ and $(\mathbf{B}_i)_{M\leq i\leq N}$ be the following two processes:

$$\mathbf{A}_i = (\mathbf{A}_{i,1},\dots,\mathbf{A}_{i,d_i^A})^T = (A^i_{i-M+1}, A^i_{i-M+2},\dots, A^i_{\min\{i-1,N-M+1\}}, A^i_i)^T$$

and

$$\mathbf{B}_i = (\mathbf{B}_{i,1},\dots,\mathbf{B}_{i,d_i^B})^T = (A^i_{i-M+2}, A^i_{i-M+3},\dots, A^i_{\min\{i-1,N-M+1\}}, A^i_i)^T.$$

In the previous equations, $A^{i_2}_{i_1}$, with i_1 and i_2 two natural numbers, denotes a quantity closely related to the quantity introduced in Equation (1)

$$A^{i_2}_{i_1} := \frac{1}{i_2-i_1+1}\sum_{j=i_1}^{i_2} S_{T_j}.$$

In particular, $\mathbf{B}_i$ can be obtained from $\mathbf{A}_i$ by dropping the first component $\mathbf{A}_{i,1}$. Let $\mathcal{V}_i$ be the option value at time T_i, which is determined by the process of partial averages $\mathbf{A}_i$ in the following way

$$\mathcal{V}_i(\mathbf{A}_i) = \max(\Psi_i^{\mathbf{A}}(\mathbf{A}_i), C_i(\mathbf{B}_i)),$$

with C_i the continuation value function of the moving average option at time T_i, where we explicitly write the dependence upon the process $\mathbf{B}_i$. The latter is given by the following relation:

$$C_i(\mathbf{B}_i) := \mathbb{E}_{T_i,\mathbf{B}_i}\left[e^{-r\Delta t}\mathcal{V}_{i+1}(\mathbf{A}_{i+1})\right],$$

where $\mathbb{E}_{T_i,\mathbf{B}_i}$ represents the expectation at time T_i given that $\mathbf{B}_i$ is the value of the process $\mathbf{B}$ at time T_i. Moreover, $\Psi^{\mathbf{A}}_{T_i}$ denotes the payoff as a function of the process $\mathbf{A}_i$

$$\Psi^{\mathbf{A}}_{T_i}(\mathbf{A}_i) := \max(0, A^i_i - A^i_{i-M+1}).$$

Then, the dynamic programming problem of the function $\mathcal{C}$ is given by (cfr. Equation (4.10) of Goudenège et al. (2022)):

$$\begin{cases} C_N(\mathbf{B}_N) = 0, \\ C_{N-1}(\mathbf{B}_{N-1}) = \mathrm{Call}\left(t_{N-1}, t_N, A^{N-1}_{N-1}, \frac{MA^i_{i-M+1}-A^i_i}{M-1}\right), \\ C_i(\mathbf{B}_i) = \mathbb{E}_{T_i,\mathbf{B}_i}\left[e^{-r\Delta t}\max\left(\Psi^A_{i+1}(\mathbf{A}_{i+1}), C_{i+1}(\mathbf{B}_{i+1})\right)\right], \end{cases}$$

where $N-2\leq i\leq M$ and $\mathrm{Call}(t_0,T,S_0,K)$ is the price of a European call option on S with inception time t_0, maturity T, spot value S_0, and strike K. At this point, Goudenège et al. (2022)

exploit GPR to learn C_i from a few observed values. Toward this aim, they consider a set X^i of P points whose elements are the simulated values for $\mathbf{B}_i$. More precisely:

$$X^i = \{\mathbf{x}^{i,p} = (x_1^{i,p}, \dots, x_{d_i^B}^{i,p}),\ p = 1, \dots, P\} \subseteq \mathbb{R}^{d_i}.$$

The GPR-GHQ method determines $C_i(\mathbf{x}^{i,p})$ for each $\mathbf{x}^{i,p} \in X^i$ through Q-points GHQ by exploiting the following relation

$$C_i^{GHQ}(\mathbf{x}^{i,p}) = e^{-r\Delta t} \sum_{q=1}^{Q} w_q \max\left(\Psi_{i+1}^A(\tilde{x}_0^{i,p,q}, \tilde{\mathbf{x}}^{i,p,q}), \mathcal{C}_{i+1}(\tilde{\mathbf{x}}^{i,p,q})\right),$$

where $\tilde{x}_{d_{i+1}}^{i,p,q} = S^{i,p,q}$, $\tilde{x}_i^{n,p,q} = \frac{(M-i-1)x_{i+1}^{n,q} + S^{n,p,q}}{M-i}$, and $\tilde{x}_0^{n,p,q} = \frac{(M-1)x_1^{n,q} + S^{n,p,q}}{M}$. In particular, the vector $(\tilde{x}_0^{i,p,q}, \tilde{\mathbf{x}}^{i,p,q})$ is a possible outcome for $\mathbf{A}_{i+1}|\mathbf{B}_i = \mathbf{x}^{i,p}$. The above equation, can be evaluated only if the quantities $C_{i+1}(\tilde{\mathbf{x}}^{i,p,q})$ are known for all the future points $\tilde{\mathbf{x}}^{i,p,q}$. In order to do so, Goudenège et al. (2022) employ the GPR method by leading to the subsequent equation:

$$C_i^{GPR-GHQ}(\mathbf{x}^{i,p}) = e^{-r\Delta t} \sum_{q=1}^{Q} w_q \max\left(\Psi_{i+1}^A(\tilde{x}_0^{i,p,q}, \tilde{\mathbf{x}}^{i,p,q}), C_{i+1}^{GPR}(\tilde{\mathbf{x}}^{n,p,q})\right),\ p \in \{1, \dots, P\}. \tag{23}$$

Once the function $C_{T_i}^{GPR-GHQ}$ has been estimated, the option price $\mathcal{V}_0$ at inception can be computed by discounting the expected option value at time T_M, which is the first time step the option can be exercised; Goudenège et al. (2022) employ a Monte Carlo approach with antithetic variables to estimate the expected value involved in the latter step.

A.2 Details on the binomial Markov chain method

Now, we describe the binomial chain. Goudenège et al. (2022) consider a Markov chain defined on a Cox-Ross-Rubinstein (CRR, henceforth) binomial tree. The set of all possible states at time T_i for $M \leq i$ is given by

$$\{(\mathbf{s}_p, S_{T_i,k}),\ p = 1, \dots, 2^{M-1},\ k = 0, \dots, i\},$$

where $\mathbf{s}_p = (\mathbf{s}_{p,1}, \dots, \mathbf{s}_{p,M-1})^T$ and $S_{T_i,k} = S_0 e^{(2k-i)\sigma\sqrt{\Delta t}}$. In particular, the payoff for a state $(\mathbf{s}_p, S_{T_i,k})$ becomes:

$$\Psi^{CRR}(\mathbf{s}_p, S_{T_i,k}) = \max\left(S_{T_i,k} - \frac{1}{M}\sum_{j=1}^{M} S_{j,k} \exp\left(-\sigma\sqrt{\Delta t} \sum_{\ell=M-j}^{M-1} (2\mathbf{s}_{p,\ell} - 1)\right)\right). \tag{24}$$

In particular, if the process state at time T_i is $(\mathbf{s}_p, S_{T_i,k})$, then the possible next states are denoted with $(\mathbf{s}_p^{\text{up}}, S_{T_i,k}^{\text{up}})$ and $(\mathbf{s}_p^{\text{up}}, S_{T_i,k}^{\text{dw}})$; option evaluation is performed by moving backward along the tree. By exploiting the positive homogeneity of the continuation value, denoted by $C_{T_i}^{CRR}$, Goudenège et al. (2022) arrive at the following recursive formula:

$$\begin{cases} C_N^{BC}(\mathbf{s}_p) = 0, \\ C_N^{BC}(\mathbf{s}_p) = e^{-r\Delta t}\left[p_{\text{up}} e^{\sigma\sqrt{\Delta t}} \max(\Psi^{CRR}(\mathbf{s}_p^{\text{up}}, 1), \mathcal{C}_{i+1}^{BC}(\mathbf{s}_p^{\text{up}})) \right. \\ \qquad\qquad \left. + p_{\text{dw}} e^{-\sigma\sqrt{\Delta t}} \max(\Psi^{CRR}(\mathbf{s}_p^{\text{dw}}, 1), C_{i+1}^{BC}(\mathbf{s}_p^{\text{dw}}))\right]. \end{cases} \tag{25}$$

Finally, once the continuation value is available at time step M the option value at inception is obtained by averaging the option value at the various states of the binomial chain at time T_M. The total number of possible states is $O(N2^M)$ and the computational cost is $O(N2^{M+1})$.

B Signatures methods

This section collects some basic concepts and definitions on signatures. In particular, Subsection B.1 provides background material on signatures, with an adaptation of the signature transform to the space of streams of data, and Subsection B.2 discusses some numerical aspects of signatures.

B.1 Background material on signatures

We start with the definition of the signature of a path. Let $X : J \to \mathbb{R}^d$ be a d-dimensional path where J is a compact interval. A path X is of finite p-variation for certain $p \geq 1$ if the p-variation of X, defined by

$$\|X\|_{p,J} = \left(\sup_{D_J \subset J} \sum_i \|X_{T_i} - X_{T_{i-1}}\|^p \right)^{1/p}$$

is finite. In the previous equation, the supremum is taken over all possible finite partitions $D_J = \{T_i\}_i$ of the interval J. We let $\mathcal{V}^p(J, E)$, with $E \subseteq \mathbb{R}^d$, be the set of any continuous path $X : J \to E$ of finite p-variation.

Definition 4 (The signature of a path)**.** Let J be a compact interval and $X \in \mathcal{V}^p(J, E)$ such that the integration in Equation (26) makes sense. The signature $S(X)$ of X over the time interval J is defined as follows

$$S(X) = \mathbf{X}_J = (1, X^1, \ldots, X^n, \ldots),$$

where for each integer $n \geq 1$,

$$X^n = \left(\int \ldots \int_{\substack{u_1 < \ldots < u_n \\ u_1, \ldots, u_n \in J}} dX_{u_1} \otimes \ldots \otimes dX_{u_n} \right) \in E^{\otimes n}, \tag{26}$$

with $\otimes$ denoting the tensor product.

Let us now take $E = \mathbb{R}^d$ for simplicity. We notice that the $n = 0$ term is $1 \in (\mathbb{R}^d)^{\otimes 0} = \mathbb{R}$, the first term belongs to $\mathbb{R}^d$, the second term belongs to $\mathbb{R}^d \otimes \mathbb{R}$ (that is, the space of matrices of size $d \times d$), and, in general, the k^{th} term belongs to $(\mathbb{R}^d)^{\otimes k} = \mathbb{R}^d \otimes \ldots \otimes \mathbb{R}^d$, k times (that is, the space of tensors of shape $(d, \ldots, d)$, k times. In particular, it turns out that the signature of X is an element of the tensor algebra space $T((\mathbb{R}^d))$, which is also the "free" algebra on $\mathbb{R}^d$, defined as

$$T((\mathbb{R}^d)) := \left\{ (a_0, a_1, \ldots, a_n, \ldots) \,|\, \forall n \geq 0,\, a_n \in (\mathbb{R}^d)^{\otimes n} \right\}. \tag{27}$$

We notice that (27) can also be written as

$$T((\mathbb{R}^d)) := \{ a \,|\, a = \sum_{k=0}^{+\infty} \sum_{i_1, \ldots, i_k = 1}^{d} a_{i_1 \ldots i_k} e_{i_1} \cdots e_{i_k} \},$$

where $e_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ with 1 in the i^{th} position, are elements of the basis of $\mathbb{R}^d$.

B.2 Numerical aspects of signatures

The truncated signature of X of order n is denoted by $S^n(X)$, i.e., $S^n(X) = (1, X^1, \ldots, X^n)$, for every integer $n \geq 1$. The truncated signatures of X of order n lies in $T^n((\mathbb{R}^d)) := \bigotimes_{i=0}^{n} E^{\otimes i}$. In particular, the signature truncated at order n is a vector of dimension

$$s_d(n) = \sum_{k=0}^{n} d^k = \frac{d^{n+1} - 1}{d - 1}, \tag{28}$$

if $d \geq 2$, $s_d(n) = n + 1$ if $d = 1$. The Python software `iisignature` that we used for our calculations ignores the constant number 1 (first element of any signature) and, thus, the dimension reduces to $s_d(n) - 1$.

Given a path $X \in \mathcal{V}^p(J, E)$, with $p \geq 1$, we know that the signature of the path uniquely defines (up to tree-like equivalence) the path itself. This explains why it can be so convenient to "summarize" a path in terms of its signatures coefficients: essentially, no information is lost. This result for paths of bounded variation is due to Hambly et al. (2010) and has been extended by Boedihardjo et al. (2016) to geometric rough paths, to which we refer for the definition of tree-like equivalence as well.

Theorem B.1 (Uniqueness of Signature)**.** *Let $X \in \mathcal{V}^p([0,T], \mathbb{R}^d)$, $p \geq 1$. Then $S(X)$ determines X up to the tree-like equivalence.*

To avoid having degenerate paths (in a tree-like sense), it has become customary to define the so-called time-augmented path version of a path by $\widehat{X}_t = (t, X_t)$, which is a path in $\mathbb{R} \times E$. Indeed, the following proposition holds true.

Proposition B.2 (Uniqueness of signatures)**.** *Let $X \in \mathcal{V}^p(J, E)$ and $\widehat{X}$ its time-augmented version. Then $S(\widehat{X})$ uniquely determines X up to translation.*

A relevant property of signatures is given by the following Proposition, which states that the terms of the signature decay in size in a factorial way; see Lemma 2.1.1 in T. J. Lyons (1998).

Proposition B.3 (Decay of signature terms)**.** *Let $X \in \mathcal{V}^p(J, E)$. Then*

$$\|X^n\| \leq \frac{C(X)^n}{n!},$$

where X^n is defined in Equation (26), $C(X)$ is a constant depending on X and $\|\cdot\|$ is any tensor norm on $(\mathbb{R}^d)^{\otimes n}$.

Furthermore, we have to cite another relevant property of signatures, namely that the signature is invariant to time reparametrization; see T. J. Lyons (1998).

Proposition B.4 (Invariance to time reparametrization)**.** *Let $X \in \mathcal{V}^p(J, E)$. Let $\psi : E \to E$ be continuously differentiable, increasing, and surjective. Then $S(X) = S(X \circ \psi)$, where "$\circ$" denotes the composition.*

In practice, we interpret a stream of data as a discretization of a path. The space of streams of data is defined as:

$$\mathrm{Str}(E) = \{\boldsymbol{x} = (x_1, x_2, \ldots, x_N) \,:\, x_i \in \mathbb{R}^d,\, N \in \mathbb{N}\}.$$

Given $\boldsymbol{x} \in \mathrm{Str}(E)$, the integer N is called the length of $\boldsymbol{x}$. Furthermore, for $a, b \in \mathbb{R}$ such that $a < b$, fix

$$a = u_1 < u_2 < \ldots < u_{N-1} < u_N = b.$$

Let $J = [a, b]$ and $X = (X^1, \ldots, X^d) : J \to E$ be in $\mathcal{V}^p(J, E)$ such that $X_{u_i} = x_i$ for all i, and interpolated in continuously differentiable, increasing, and surjective way on the intervals in between. Then X is called interpolation of $\boldsymbol{x}$. We have the following definition.

Definition 5 (Signature of the interpolating path)**.** Let $\boldsymbol{x} \in \mathrm{Str}(E)$. Let X be an interpolation of $\boldsymbol{x}$. Then the signature of $\boldsymbol{x}$ is defined as $S(\boldsymbol{x}) = S(X)$ and similarly the truncated signature.

In particular, thanks to Proposition B.4 the previous definition is well defined and independent of the interpolation choice.

Finally, one of the most important properties of signatures is being universal approximators, that is continuous functions on paths can be approximated on compact spaces by linear functions on the signature of the path; see, for instance, Király et al. (2019).

Theorem B.5 (Universal approximators)**.** *Let K be a compact set in $\mathcal{V}^p([0,T],\mathbb{R}^d)$ and $f \in \mathcal{C}(K,\mathbb{R}^d)$ be an $\mathbb{R}^d$-valued continuous function on such compact. Then, for any $\epsilon > 0$, there exists a linear function l such that* $\sup_{X\in K} \|f(X) - \langle l, S(\widehat{X})\rangle\| < \epsilon$.

C Snowball and lock-in payoffs

A certificate is a financial instrument issued by a financial intermediary that allows to take a position on one or many underlying assets. Here, we focus on certificates with maturity date T written on a single traded assets, whose price process is S_t, and paying coupons γ_i on payment dates T_i with $1 \leq i \leq N$, $T_1 > 0$ and $T_N := T$. On the certificate maturity date the whole principal amount, or a fraction of it, is redeemed according to the performance of the underlying asset. The principal-amount redemption is defined as

$$\phi(s,H) := \mathbb{1}_{\{P(s)>H\}} + \mathbb{1}_{\{P(s)<H\}} P(s), \tag{29}$$

where P is the basket worst performance and it is given by

$$P(s) := \frac{s}{s_0} \tag{30}$$

with respect to a level s_0 usually read from the market before issuing the certificate.

In case the contract is terminated before the maturity the whole principal amount is redeemed. This case may happen if the contract contains early-termination features, as discussed in the next section. For later convenience, we define the principal-amount redemption on maturity date or on early-termination date as following

$$\Psi^{\mathrm{C}}_{T_i}(N;H) := 1 - \mathbb{1}_{\{i=N\}} \left(1 - \phi(S_{T_N}, H)\right). \tag{31}$$

The coupon amounts paid by the certificate can be fixed at inception or they can depend on the asset performances. We wish to investigate the latter case, and, in particular, the case of coupons with "snowball" and "lock-in" features.

We start by discussing the "snowball" version. At each payment date the certificate pays a coupon only if the basket worst performance is above a barrier level K. When the payment occurs, the quantity of cash paid is equal to the sum of the cash flows not yet paid. We can write this feature in the following way. First, we introduce the set $\mathcal{J}(i)$ of indices less than i such that a coupon can be paid on the corresponding payment date without taking into account if the product is early terminated. In formulae we have

$$\mathcal{J}(i) := \{j \in \{1,\dots,i-1\} : P(S_{t_j}) > K\}. \tag{32}$$

We can now define the coupon as

$$\gamma_i := \mathbb{1}_{\{P(S_{t_i})>K\}} \sum_{j=1+\sup \mathcal{J}(i)}^{i} c_j \tag{33}$$

with the convention that the supremum of $\mathcal{J}(i)$ is 0 when the set is empty. In the previous equation $\{c_1, \ldots, c_N\}$ is a pre-determined set of non-negative cash flows.

Then, we discuss the "lock-in" version. At each payment date the certificate pays a coupon only if the basket worst performance at such date, or at any of the previous payment dates, is above a barrier level K. We can define the coupon as

$$\gamma_i := \max_{j\in[1,i]} \mathbb{1}_{\{P(S_{t_j})>K\}}\, C_{T_i}. \tag{34}$$

We can evaluate certificates with the possibility of an early exercise by the issuer. Again, we consider that the certificate can be exercised on any date in the time grid $\{T_1, \ldots, T_N\}$ previously defined. We setup the pricing problem by starting from the last exercise date T_N, and we introduce the $\mathcal{F}_{T_N}$-measurable value function V_N^{C}, defined as

$$V_{T_N}^{\mathrm{C}} := \Psi_{T_N}^{\mathrm{C}}. \tag{35}$$

On the previous dates T_i, with $0 < i < N$ the option buyer may decide to exercise the option, by choosing the minimum between the immediate redemption of the principal amount and the continuation value of the contract. Thus, we can write

$$V_{T_i}^{\mathrm{C}} := \gamma_i + \min\left\{\Psi_{T_i}^{\mathrm{C}}, B_{T_i}\,\mathbb{E}\left[\left.\frac{V_{T_{i+1}}^{\mathrm{C}}}{B_{T_{i+1}}}\right|\mathcal{F}_{T_i}\right]\right\}, \tag{36}$$

The price of the certificate can be calculated by applying the above recursion up to T_1 and by calculating the certificate price as given by

$$V_0^{\mathrm{C}} := \mathbb{E}\left[\frac{V_{T_1}^{\mathrm{C}}}{B_{T_1}}\right]. \tag{37}$$

D Technical insights on sensitivity computations

Here, we describe in more details the problem of dealing with discontinuous sensitivities. In particular, we expand the discussion on the Chebyshev method presented in Section 3.4.3 in the main text of the paper. Then, we conclude this section by presenting more numerical evidences supporting our choice for the Chebyshev method.

D.1 Discontinuous sensitivities in the Chebyshev method

In the case of American put options the buyer may exercise the option on contract inception. As a consequence the function $S \mapsto V(S)$ has a discontinuous second derivative. In our specific settings of positive interest rates and zero dividend yield (see Section 4.1), we know that the immediate exercise occurs when the spot price is less than a critical value $S^\star$, which depends on market and contract data. Moreover, when it occurs, the option price is equal to its intrinsic value, a linear function of the underlying spot price, leading to a Gamma equal to zero.

We should perform the numerical calculation only when the spot price is greater than $S^\star$. In the case of the Chebyshev method we can achieve this by redefining the grid of spot prices, previously defined in (18), as

$$S_\epsilon^\ell := \max\left\{S, \frac{S^\star}{1-\epsilon}\right\}\left(1 + \epsilon\cos\left(\frac{\ell\pi}{N_c - 1}\right)\right), \tag{38}$$

where $S > S^\star$ is the market spot price.

The above derivation requires the knowledge of $S^\star$. However, if the value of $S^\star$ is not known in advance, we can implement a simple heuristic.

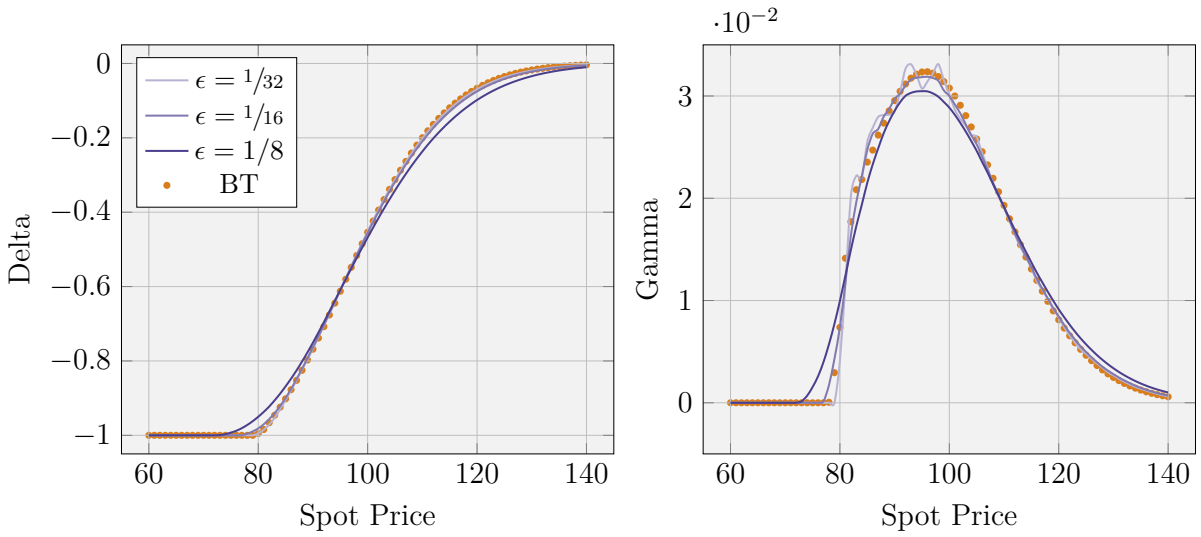

Figure 3: Delta and Gamma risk matrices for American put options with maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. Orange dots are calculated with a standard binomial tree implementation, while blue lines with the finite difference method.

1. We choose N_c to be odd, namely $N_c := 2K_c + 1$ for a positive integer K_c, and we set $S^\star := S(1-\epsilon)$ in equation (38), so that $S_\epsilon^{K_c} = S$.
2. We calculate the corresponding Monte Carlo prices and we check if on $S_\epsilon^{K_c}$ an immediate exercise occurs. If it is the case we return zero for the Gamma, otherwise we go on.
3. We check if an immediate exercise occurs for some S_ϵ^ℓ with $\ell < K_c$. If this is the case we discard such points from the interpolator training set.
4. We train the interpolator on the remaining points, and we calculate the sensitivity.

D.2 Numerical investigations on American put sensitivities

We start by presenting the results in Figure 3 Delta and Gamma in the case of the finite difference method for American put options at different spot prices. We can see that increasing ϵ the noise is reduced, but a bias in the sensitivity arises, since only in the limit of vanishing ϵ we recover the true derivative values.

Then, we continue with the regression method and we show in Figure 4 the calculation of Gamma for American put options at different spot prices by varying both ϵ and B. We can see that a delicate fine tuning is required to properly select these variables to reduce noise without introducing a bias. In particular, we notice that higher terms in the Taylor's approximation are required to correctly learn the dependency of the derivative price with respect to the spot price, but they increase the sensitivity noise. In order to reduce the noise we can increase the variance of the spot-price density, but in this way we introduce a bias. Moreover, we can see that the discontinuity of Gamma cannot be reproduced with this method, since the approximating form for the derivative price is continuous in the spot price.

Finally, we analyze the Chebyshev method and we present in Figure 5 the calculation of Gamma for American put options at different spot prices by varying ϵ. We can see that, by increasing the range of spot prices used to train the interpolator, the noise decreases without generating biases.

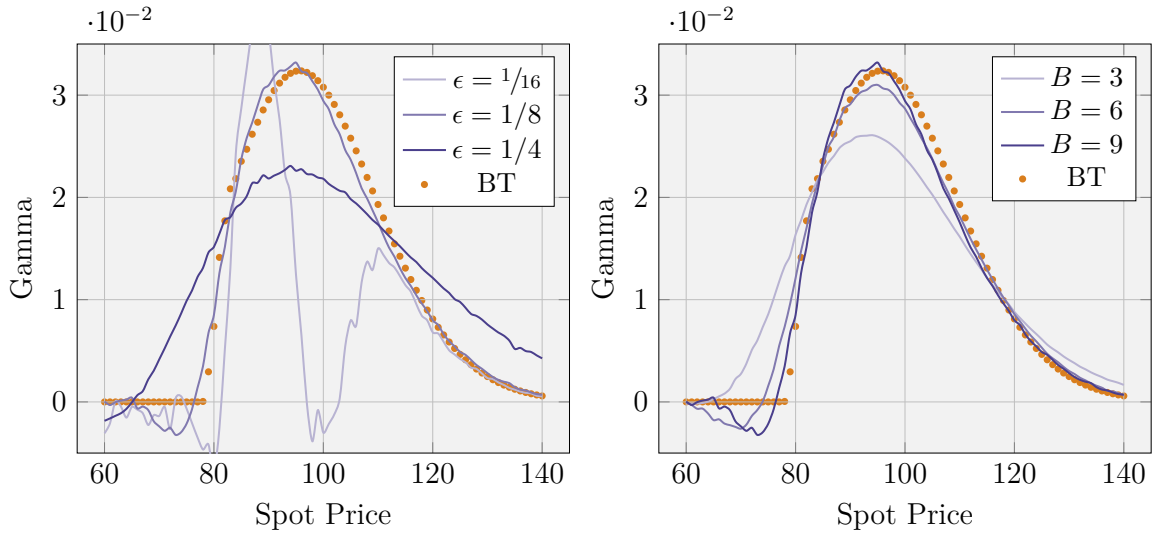

Figure 4: Gamma risk matrices for American put options with maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. Orange dots are calculated with a standard binomial tree implementation, while blue lines with the regression method. Left panel: different choices of ϵ for $B = 9$. Right panel: different choices of B for $\epsilon = 1/8$.

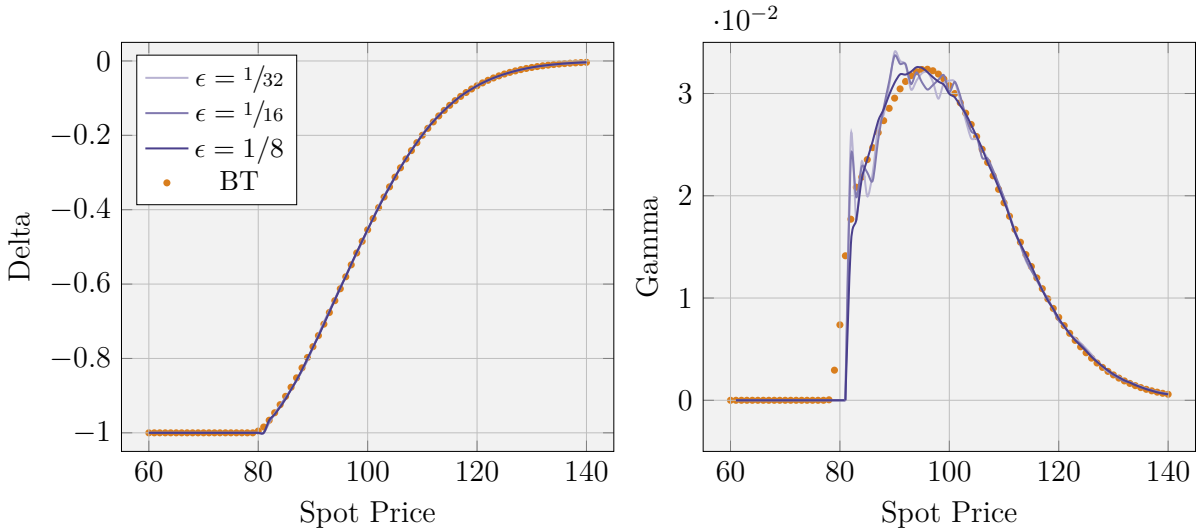

Figure 5: Delta and Gamma risk matrices for American put options with maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. Orange dots are calculated with a standard binomial tree implementation, while blue lines with the Chebyshev method.

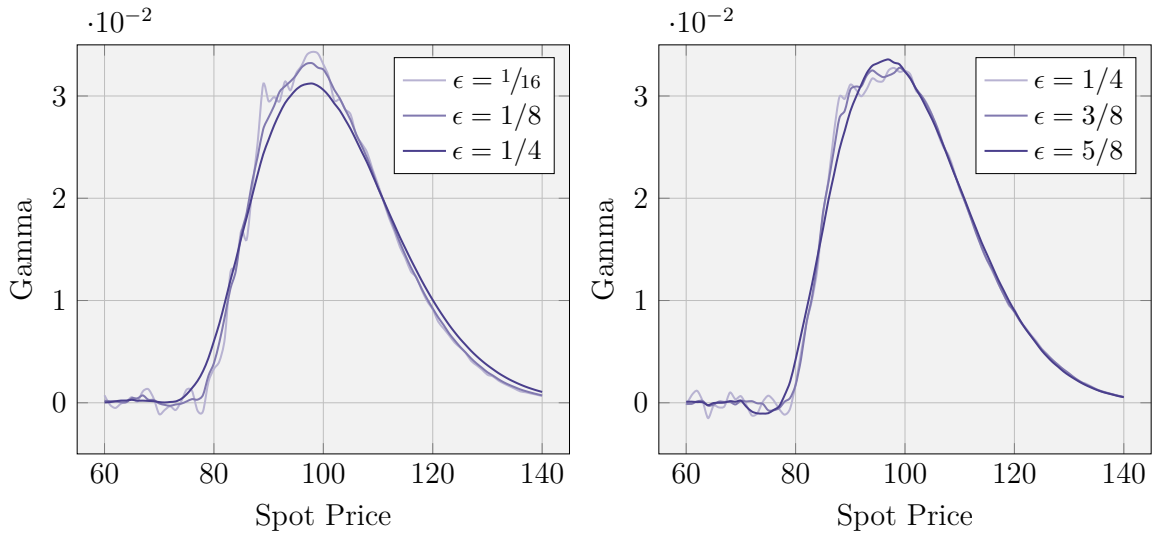

Figure 6: Gamma risk matrices for fixed-strike Asian options with moving window of $M = 2$ days, maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. LSMC method with polynomial basis functions. Left panel: finite-difference method with different choices of ϵ. Right panel: Chebyshev method with different choices of ϵ.

D.3 Numerical investigations on fixed-strike Asian sensitivities

We continue the analysis on sensitivity computations by going beyond the simple, but relevant, case of American options. We find that the Chebyshev method seems a good candidate as target method. Anyway, we prefer to check again our findings also in the case of other payoffs.

In the case of fixed-strike asian payoffs we can see in Figure 6 the behavior of Gamma sensitivity for a moving window of length $M = 2$ with different choices of ϵ when using polynomials for the regression. We obtain the same pattern depicted in the case of American put options. The finite-difference method stabilizes as we increase ϵ, but a visible bias arises, as we can see by comparing the two panels of the Figure.

For the same fixed-strike Asian payoff we repeat the exercise with R-FFNNs as basis functions. We obtain a very similar result, and we do not notice an increase in noise.

D.4 Numerical investigations on certificate sensitivities

We continue the analysis on sensitivity computations with the certificate case. We start with the snowball version, whose characteristics are described in Section 6.2. We show the results for different regression basis functions in Figure 8 and 9. Polynomials basis function are taken up to third order; random neural networks have 120 inner nodes; signature methods are truncated at third degree with lead-lag and time embedding. Again we can see that the Chebyshev method, although a little noisier than before, does not show the bias of the finite difference method, which is smoother, but we can see from the diagrams that it moves away from the other curves as soon as the shift ϵ increases.

We continue with the lock-in version, whose characteristics are described in Section 6.3. We show the results for different regression basis functions in Figure 10 and 11. Polynomials basis function are taken up to third order; random neural networks have 120 inner nodes; signature methods are truncated at third degree with lead-lag and time embedding. We can see that the finite-difference method is quite biased, while the Chebyshev method seems working well.

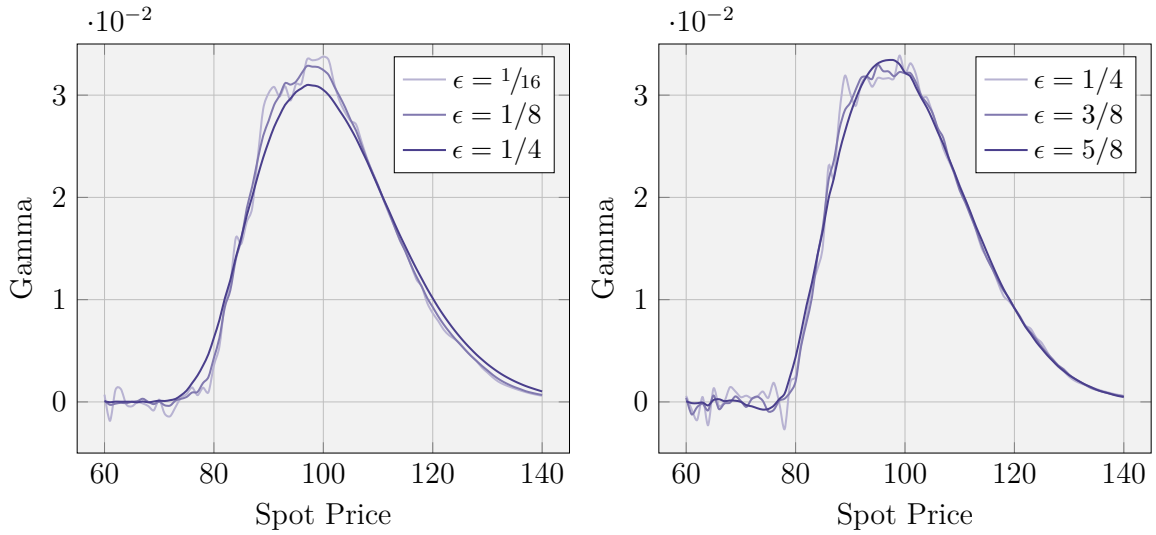

Figure 7: Gamma risk matrices for fixed-strike Asian options with moving window of $M = 2$ days, maturity $T = 0.2$, strike $K = 100$ and time steps $N = 50$. LSMC method with R-FFNN basis functions. Left panel: finite-difference method with different choices of ϵ. Right panel: Chebyshev method with different choices of ϵ.

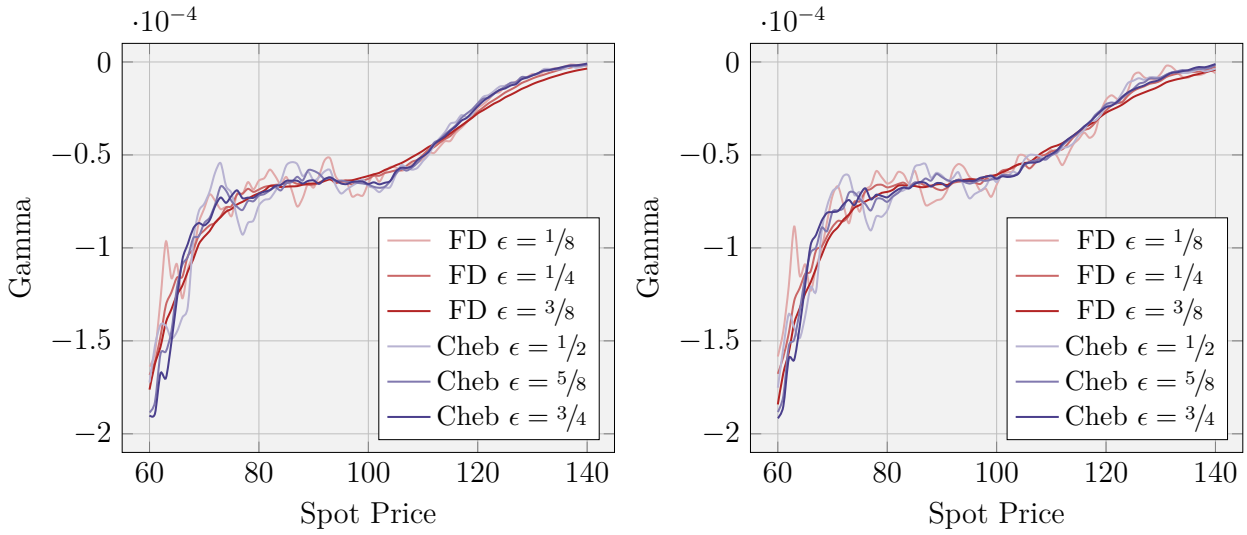

Figure 8: Gamma risk matrices for snowball certificates, maturity $T = 2$. LSMC method with polynomial basis functions (left panel), R-FFNN basis functions (right panel). Finite difference method in red, Chebyshev method in blue.

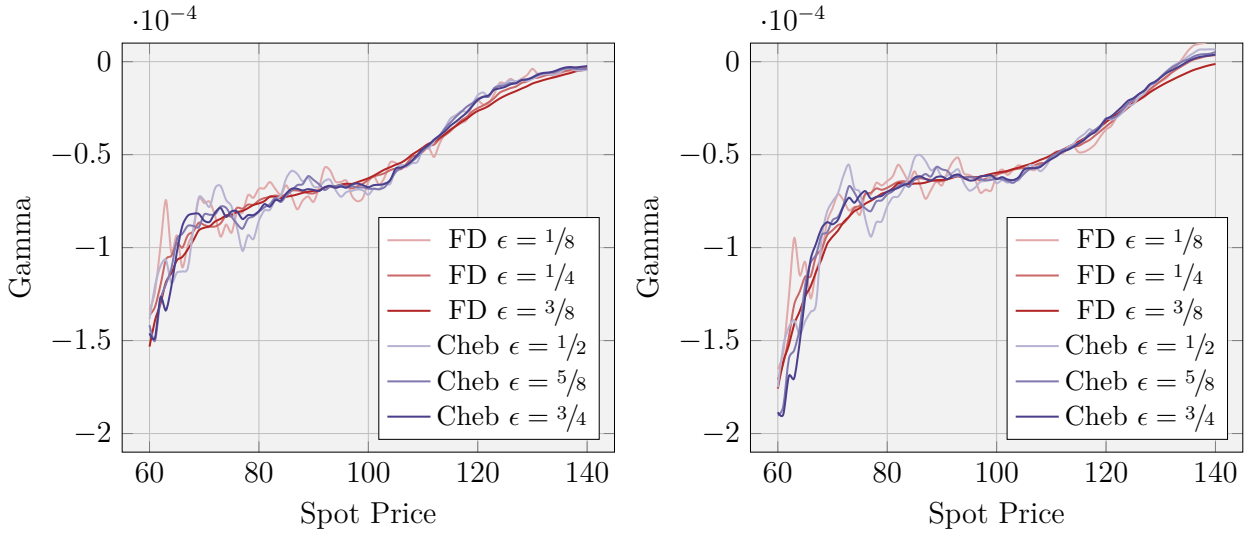

Figure 9: Gamma risk matrices for snowball certificates, maturity $T = 2$. LSMC method with R-RNN basis functions (left panel), signature basis functions (right panel). Finite difference method red, Chebyshev method in blue.

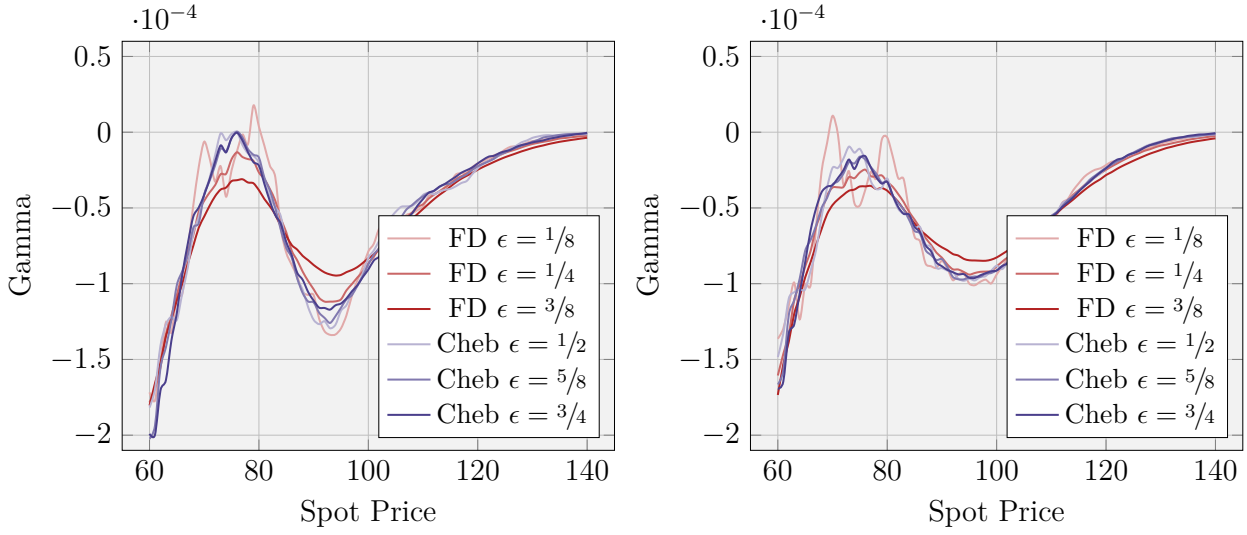

Figure 10: Gamma risk matrices for lock-in certificates, maturity $T = 2$. LSMC method with polynomial basis functions (left panel), R-FFNN basis functions (right panel). Finite difference method in red, Chebyshev method in blue.

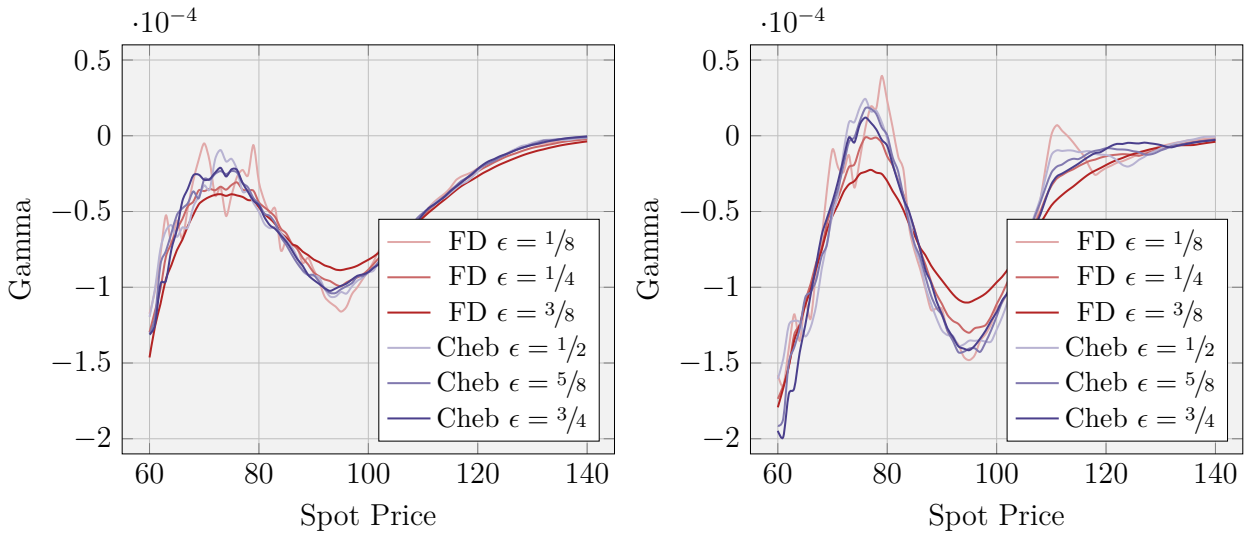

Figure 11: Gamma risk matrices for lock-in certificates, maturity $T = 2$. LSMC method with R-RNN basis functions (left panel), signature basis functions (right panel). Finite difference method red, Chebyshev method in blue.

E Supplemental material

We collect in this section further numerical results, which support the discussion of the main body of the paper. In Tables 17, 18, 19, 20 and 21 we report the statistical uncertainties of Monte Carlo simulations used to price all the contracts discussed in the main body of the paper. Statistical uncertainties are reported at one-sigma confidence level.

Floating-strike Asian option: statistical uncertainties							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
Polynomial, $\rho = 2, \deg = 2$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
Polynomial, $\rho = 3, \deg = 2$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
Polynomial, $\rho = 4, \deg = 2$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 1, h = 10$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 10$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 10$	0.001	0.002	0.002	0.002	0.004	0.005	0.007
R-FFNN, $\rho = 4, h = 10$	0.001	0.002	0.002	0.002	0.004	0.005	0.006
R-FFNN, $\rho = 1, h = 40$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 40$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 40$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 4, h = 40$	0.001	0.002	0.002	0.002	0.004	0.005	0.006
R-FFNN, $\rho = 1, h = 120$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 120$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 120$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 4, h = 120$	0.001	0.002	0.002	0.002	0.004	0.005	0.007
R-FFNN, $\rho = 1, h = 250$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 250$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 250$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 4, h = 250$	0.001	0.002	0.002	0.002	0.004	0.005	0.008
R-RNN, $\rho = 4, h = 10$	0.001	0.002	0.002	0.003	0.004	0.007	0.009
R-RNN, $\rho = 4, h = 40$	0.001	0.002	0.002	0.003	0.004	0.007	0.005
R-RNN, $\rho = 4, h = 120$	0.001	0.002	0.002	0.003	0.004	0.007	0.009
R-RNN, $\rho = 4, h = 150$	0.001	0.002	0.002	0.003	0.004	0.007	0.009
Signature, $\rho = 4, n = 2$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
Signature, $\rho = 4, n = 3$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
Signature, $\rho = 4, n = 5$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=True	0.001	0.002	0.002	0.002	0.004	0.006	0.007
RandSig, $\varsigma = 0.05, k^\star = 20$, norm=True	0.001	0.002	0.002	0.002	0.004	0.006	0.008
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=True	0.001	0.002	0.002	0.002	0.004	0.006	0.008
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=False	0.001	0.002	0.002	0.002	0.004	0.006	0.007
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=False	0.001	0.002	0.002	0.002	0.004	0.005	0.007
RandSig, $\varsigma = 0.3, k^\star = 10$, norm=False	0.001	0.002	0.002	0.002	0.003	0.005	0.007

Table 17: Statistical uncertainties of the Monte Carlo simulation used to price floating-strike American-style Asian options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). Benchmark models: GPR-GHQ and binomial Markov chain of Goudenège et al. (2022), Bernhart et al. (2011), Lelong (2019). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature and randomized signature (RandSig) based methods.

Fixed-strike Asian option: statistical uncertainties							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	0.011	0.011	0.011	0.115	0.011	0.010	0.009
Polynomial, $\rho = 2, \deg = 2$	0.011	0.012	0.012	0.012	0.011	0.011	0.010
Polynomial, $\rho = 3, \deg = 2$	0.011	0.012	0.012	0.012	0.011	0.011	0.010
Polynomial, $\rho = 4, \deg = 2$	0.011	0.011	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 1, h = 10$	0.011	0.011	0.011	0.011	0.011	0.010	0.009
R-FFNN, $\rho = 2, h = 10$	0.011	0.012	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 3, h = 10$	0.011	0.012	0.012	0.012	0.011	0.010	0.010
R-FFNN, $\rho = 4, h = 10$	0.006	0.007	0.007	0.008	0.010	0.010	0.010
R-FFNN, $\rho = 1, h = 40$	0.011	0.011	0.011	0.011	0.011	0.010	0.009
R-FFNN, $\rho = 2, h = 40$	0.011	0.012	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 3, h = 40$	0.011	0.012	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 4, h = 40$	0.008	0.008	0.009	0.009	0.010	0.010	0.010
R-FFNN, $\rho = 1, h = 120$	0.011	0.011	0.011	0.011	0.011	0.010	0.009
R-FFNN, $\rho = 2, h = 120$	0.011	0.011	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 3, h = 120$	0.011	0.012	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 4, h = 120$	0.010	0.011	0.011	0.011	0.011	0.011	0.010
R-FFNN, $\rho = 1, h = 250$	0.011	0.011	0.011	0.011	0.011	0.010	0.009
R-FFNN, $\rho = 2, h = 250$	0.011	0.011	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 3, h = 250$	0.011	0.011	0.012	0.012	0.011	0.011	0.010
R-FFNN, $\rho = 4, h = 250$	0.011	0.011	0.011	0.011	0.011	0.011	0.010
R-RNN, $\rho = 4, h = 10$	0.014	0.015	0.014	0.014	0.014	0.012	0.011
R-RNN, $\rho = 4, h = 40$	0.014	0.015	0.014	0.014	0.014	0.012	0.011
R-RNN, $\rho = 4, h = 120$	0.014	0.014	0.014	0.014	0.014	0.012	0.011
R-RNN, $\rho = 4, h = 150$	0.014	0.014	0.014	0.014	0.014	0.012	0.011
Signature, $\rho = 4, n = 2$	0.010	0.011	0.011	0.011	0.011	0.010	0.009
Signature, $\rho = 4, n = 3$	0.011	0.011	0.011	0.011	0.011	0.010	0.010
Signature, $\rho = 4, n = 3$, no lead-lag	0.011	0.011	0.011	0.011	0.011	0.010	0.010
Signature, $\rho = 4, n = 5$	0.011	0.011	0.011	0.011	0.011	0.011	0.010
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=True	0.011	0.011	0.011	0.011	0.011	0.010	0.009
RandSig, $\varsigma = 0.05, k^\star = 20$, norm=True	0.011	0.011	0.011	0.011	0.011	0.010	0.009
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=True	0.011	0.011	0.011	0.011	0.011	0.010	0.010
RandSig, $\varsigma = 0.05, k^\star = 40$, norm=False	0.011	0.011	0.011	0.011	0.011	0.010	0.009
RandSig, $\varsigma = 0.3, k^\star = 20$, norm=False	0.009	0.008	0.008	0.008	0.007	0.007	0.007
RandSig, $\varsigma = 0.3, k^\star = 10$, norm=False	0.009	0.009	0.009	0.009	0.009	0.008	0.007

Table 18: Statistical uncertainties of the Monte Carlo simulation used to price fixed-strike American-style Asian options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature and randomized signature (RandSig) based methods.

Floating-strike look-back option: statistical uncertainties							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
Polynomial, $\rho = 2, \deg = 2$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
Polynomial, $\rho = 3, \deg = 2$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
Polynomial, $\rho = 4, \deg = 2$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 1, h = 10$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 10$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 10$	0.001	0.002	0.002	0.003	0.004	0.005	0.007
R-FFNN, $\rho = 4, h = 10$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 1, h = 40$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 40$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 40$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 4, h = 40$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
R-FFNN, $\rho = 1, h = 120$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 120$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 120$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 4, h = 120$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 1, h = 250$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
R-FFNN, $\rho = 2, h = 250$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 3, h = 250$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-FFNN, $\rho = 4, h = 250$	0.001	0.002	0.002	0.003	0.004	0.006	0.007
R-RNN, $\rho = 4, h = 10$	0.001	0.003	0.002	0.003	0.004	0.007	0.009
R-RNN, $\rho = 4, h = 40$	0.001	0.002	0.002	0.003	0.004	0.007	0.009
R-RNN, $\rho = 4, h = 120$	0.001	0.002	0.002	0.003	0.004	0.007	0.009
R-RNN, $\rho = 4, h = 150$	0.001	0.002	0.002	0.003	0.004	0.007	0.009
Signature, $\rho = 4, n = 2$	0.001	0.002	0.002	0.002	0.003	0.005	0.007
Signature, $\rho = 4, n = 3$	0.001	0.002	0.002	0.002	0.004	0.006	0.007
Signature, $\rho = 4, n = 5$	0.001	0.002	0.002	0.002	0.004	0.006	0.007

Table 19: Statistical uncertainties of the Monte Carlo simulation used to price floating-strike American-style look-back options with maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

Fixed-strike look-back option: statistical uncertainties							
Model	**2**	**3**	**4**	**5**	**10**	**20**	**30**
Polynomial, $\rho = 1, \deg = 2$	0.011	0.011	0.011	0.010	0.009	0.007	0.005
Polynomial, $\rho = 2, \deg = 2$	0.011	0.011	0.011	0.011	0.010	0.008	0.006
Polynomial, $\rho = 3, \deg = 2$	0.011	0.011	0.011	0.011	0.010	0.008	0.006
Polynomial, $\rho = 4, \deg = 2$	0.011	0.011	0.011	0.011	0.010	0.008	0.006
R-FFNN, $\rho = 1, h = 10$	0.011	0.011	0.010	0.010	0.009	0.001	0.005
R-FFNN, $\rho = 2, h = 10$	0.011	0.011	0.011	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 3, h = 10$	0.011	0.011	0.011	0.011	0.001	0.008	0.006
R-FFNN, $\rho = 4, h = 10$	0.006	0.007	0.008	0.008	0.009	0.008	0.006
R-FFNN, $\rho = 1, h = 40$	0.011	0.010	0.010	0.010	0.009	0.007	0.005
R-FFNN, $\rho = 2, h = 40$	0.011	0.011	0.011	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 3, h = 40$	0.011	0.011	0.010	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 4, h = 40$	0.008	0.009	0.010	0.009	0.009	0.008	0.006
R-FFNN, $\rho = 1, h = 120$	0.010	0.010	0.010	0.010	0.009	0.007	0.005
R-FFNN, $\rho = 2, h = 120$	0.010	0.010	0.010	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 3, h = 120$	0.010	0.010	0.010	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 4, h = 120$	0.010	0.010	0.010	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 1, h = 250$	0.010	0.010	0.010	0.010	0.009	0.007	0.006
R-FFNN, $\rho = 2, h = 250$	0.010	0.010	0.010	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 3, h = 250$	0.010	0.010	0.010	0.010	0.010	0.008	0.006
R-FFNN, $\rho = 4, h = 250$	0.010	0.010	0.010	0.010	0.010	0.008	0.006
R-RNN, $\rho = 4, h = 10$	0.013	0.013	0.013	0.013	0.012	0.009	0.006
R-RNN, $\rho = 4, h = 40$	0.013	0.013	0.013	0.013	0.012	0.009	0.006
R-RNN, $\rho = 4, h = 120$	0.013	0.013	0.013	0.013	0.012	0.009	0.006
R-RNN, $\rho = 4, h = 150$	0.013	0.013	0.013	0.013	0.012	0.009	0.006
Signature, $\rho = 4, n = 2$	0.010	0.010	0.010	0.010	0.009	0.007	0.005
Signature, $\rho = 4, n = 3$	0.010	0.011	0.010	0.010	0.010	0.008	0.006
Signature, $\rho = 4, n = 5$	0.010	0.010	0.010	0.010	0.010	0.008	0.006

Table 20: Statistical uncertainties of the Monte Carlo simulation used to price fixed-strike American-style look-back options with strike $K = 100$, maturity $T = 0.2$ and time steps $N = 50$ for different lengths of the moving window. Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.

Certificate: uncert.	**Snowball**			**Lock-in**		
Model	**1y**	**2y**	**5y**	**1y**	**2y**	**5y**
Polynomial, deg = 2	0.00006	0.00011	0.00028	0.00006	0.00008	0.00019
Polynomial, deg = 3	0.00006	0.00011	0.00027	0.00006	0.00008	0.00018
Polynomial, deg = 5	0.00006	0.00011		0.00006	0.00008	
R-FFNN, $h = 10$	0.00006	0.00011	0.00030	0.00006	0.00007	0.00018
R-FFNN, $h = 50$	0.00006	0.00011	0.00030	0.00006	0.00007	0.00017
R-FFNN, $h = 120$	0.00006	0.00011	0.00029	0.00006	0.00007	0.00017
R-RNN, $h = 10$	0.00006	0.00011	0.00030	0.00006	0.00007	0.00017
R-RNN, $h = 50$	0.00006	0.00011	0.00030	0.00006	0.00007	0.00017
R-RNN, $h = 120$	0.00006	0.00011	0.00030	0.00006	0.00007	0.00017
Signature, $n = 2$	0.00006	0.00011	0.00022	0.00006	0.00007	0.00014
Signature, $n = 3$	0.00006	0.00011	0.00026	0.00006	0.00008	0.00017
Signature, $n = 5$	0.00006	0.00011	0.00025	0.00006	0.00008	0.00016

Table 21: Statistical uncertainties of the Monte Carlo simulation used to price snowball and lock-in certificates with maturity of 1, 2 and 5 years (other characteristics described in the text). Comparison between different models (see text). LSMC models: polynomial bases, randomized neural networks (R-FFNN and R-RNN), signature based methods.