Optimal stopping via randomized neural networks

Frontiers of Mathematical Finance
Vol. 3, No. 1, March 2024, pp. 31-77
doi:10.3934/fmf.2023022

OPTIMAL STOPPING VIA RANDOMIZED NEURAL NETWORKS

Calypso Herrera[1], Florian Krach[1], Pierre Ruyssen[2] and Josef Teichmann[*1]

[1]Department of Mathematics, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland

[2]Google Brain, Google Zurich, Brandschenkestrasse 110, 8002 Zürich, Switzerland

(Communicated by Xin Guo)

Abstract. This paper presents the benefits of using randomized neural networks instead of standard basis functions or deep neural networks to approximate the solutions of optimal stopping problems. The key idea is to use neural networks, where the parameters of the hidden layers are generated randomly, and only the last layer is trained, in order to approximate the continuation value. Our approaches are applicable to high dimensional problems where the existing approaches become increasingly impractical. In addition, since our approaches can be optimized using simple linear regression, they are easy to implement, and theoretical guarantees can be provided. We test our approaches for American option pricing on Black–Scholes, Heston and rough Heston models and for optimally stopping fractional Brownian motion. In all cases, our algorithms outperform the state-of-the-art and other relevant machine learning approaches in terms of computation time while achieving comparable results. Moreover, we show that they can also be used to efficiently compute Greeks of American options.

1. **Introduction.**

The optimal stopping problem consists of finding the optimal time to stop in order to maximize an expected reward. This problem is found in the areas of statistics, economics and financial mathematics. Despite significant advances, it remains one of the most challenging problems in optimization, in particular when more than one factor affects the expected reward. A common provable and widely used approach is based on Monte Carlo simulations, where the stopping decision is estimated via backward induction [64, 51], which is an (approximate) dynamic programming approach. Another provable approach is based on reinforcement learning (RL) [63, 64, 66, 49, 18]. Both approaches are based on the ordinary least squares approximation which involves choosing basis functions. There are many different sets of basis functions available that are commonly used, but, it can be difficult to choose a good set for the considered problem. Moreover, the number of basis functions often increases polynomially or even exponentially [51,

2020 *Mathematics Subject Classification.* Primary: 60G40; Secondary: 68T07.

Key words and phrases. Optimal stopping, American option pricing, least squares Monte Carlo, reinforcement learning, randomized neural networks, reservoir computing, Greeks of American options.

*Corresponding author: Josef Teichmann.

Section 2.2] in the dimension of the underlying process, making those algorithms impractical for high dimensions.

A relatively new approach consists of replacing the basis functions with a neural network and performing gradient descent instead of ordinary least squares [45, 47, 10, 11]. The big advantage is that the basis functions do not need to be chosen but are learned instead. Compared to using a polynomial basis, neural networks have the advantage of being dense in any space $L^p(\mu)$, for $1 \leq p < \infty$ and finite measure μ [41], while for polynomials this is only true under certain additional conditions on the measure [3]. Moreover, in many cases the neural network overcomes the curse of dimensionality, which means that it can easily scale to high dimensions. However, as the neural network is a non-convex function with respect to its parameters, the gradient descent does not necessarily converge to the global minimum, while this is the case for the ordinary least squares minimization. Hence, the main disadvantage of those methods is that there are no convergence guarantees without strong and unrealistic assumptions.

In this paper, we propose two neural network based algorithms to solve the optimal stopping problem for Markovian settings: a backward induction approach and a reinforcement learning approach. The idea is inspired by randomized neural networks [16, 42]. Instead of learning the parameters of all layers of the neural network, those of the hidden layers are randomly chosen and fixed, and only the parameters of the last layer are learned. Hence, the non-convex optimization problem is reduced to a convex problem that can be solved with linear regression. The hidden layers form random feature maps, which can be interpreted as random basis functions. In particular, in this paper we show that there is actually no need for complicated basis functions or a large number of basis functions. Our algorithms are based on the methods proposed by [51] (backward-induction approach) and [64] (reinforcement learning approach). The difference is that we use a randomized neural network instead of a linear combination of basis functions. However, a randomized neural network can also be interpreted as a linear combination of *random* basis functions. On the other hand, our algorithms can also be interpreted as the neural network extensions of these methods, where not the entire neural network but only the last layer is trained.

In addition, we provide a randomized recurrent neural network approach for non-Markovian settings. We compare our algorithms to the most relevant baselines in terms of accuracy and computational speed in different option pricing problems. With only a fraction of trainable parameters compared to existing methods, we achieve high quality results considerably faster.

In this work we mainly focus on the well known and important application of optimal stopping strategies to compute lower bounds for American option prices[1] (as was done, e.g., by [64, 51, 20, 67, 68, 69, 70, 47]). However, we also show in Section 5 how our algorithms can be used to compute upper bounds of the American

[1]Since American option prices can be formulated as a supremum over stopping times (see Section 2.2), this naturally leads to lower bounds of their prices when approximating an optimal stopping strategy (and to the correct price when finding the optimal stopping strategy). Importantly, computing the lower bound of American option prices in this way, by actually solving the optimal stopping problem (approximately), provides a control algorithm that allows its user to make the decision whether to stop or not, to achieve (approximately) optimal outcomes. On the contrary, the dual formulation of American option prices as an infimum over martingales [55], which naturally leads to upper bounds for those prices, is not an optimal stopping problem, and therefore their solution does not provide a strategy to achieve (nearly) optimal outcomes.

option prices without additional costs via the dual approach introduced in [55] and refined by [10]. Moreover, we show in Section 7.6 that our algorithms can be used to efficiently compute the Greeks of American options.

Finally, we note that our approach is very generic in the sense that it can be applied to any possible type of optimal stopping problem as long as one has access to a sampling method for paths of the underlying process (which should be stopped optimally). In particular, in the case of American option pricing, this means that our approach can be applied to any type of underlying stock model, with or without (discrete or continuous) dividends; with positive, negative or stochastic interest rates; and with any type of payoff, no matter its complexity.

2. **Optimal stopping via randomized neural networks.**

One of the most popular and most studied applications of optimal stopping is the pricing of American options. Hence, we explain our approach in this context.

2.1. **American and Bermudan options.**

An American option gives the holder the right but not the obligation to exercise the option associated with a non-negative payoff function g at any time up to the maturity. An American option can be approximated by a Bermudan option, which can be exercised only at some specific dates $t_0 < t_1 < t_2 < \cdots < t_N$, transforming the continuous-time problem to a discrete one. If the time grid is chosen small enough, the American option is well approximated by the Bermudan option. In the case of a Rough Heston model, the convergence rate of the Bermudan option price to the American option price was shown in [19, Theorem 4.2]. For equidistant dates, we simply write $0, 1, 2, \ldots, N$ instead of $t_0 < t_1 < t_2 < \cdots < t_N$.

2.2. **Option price and optimal stopping.**

For $d \in \mathbb{N}$, we assume having a d-dimensional Markovian stochastic process $(X_t)_{t \geq 0}$ describing the stock prices. With respect to a fixed (pricing) probability measure $\mathbb{Q}$, the (superhedging seller's) price of the discretized American option can be expressed through the Snell envelope described by

$$\begin{aligned} U_N &:= g(X_N), \\ U_n &:= \max\left(g(X_n), \mathbb{E}[\alpha\, U_{n+1} \,|\, X_n]\right), \quad 0 \leq n < N, \end{aligned} \tag{1}$$

where α is the step-wise discounting factor, and $g(X_n)$ is assumed to be square integrable for all n. Then, the (superhedging seller's) price of the option at time n is given by U_n and can equivalently be expressed as the optimal stopping problem

$$U_n = \sup\nolimits_{\tau \in \mathcal{T}_n} \mathbb{E}[\alpha^{\tau - n} g(X_\tau) \,|\, X_n], \tag{2}$$

where $\mathcal{T}_n$ is the set of all stopping times $\tau \geq n$. The smallest optimal stopping time is given by

$$\begin{aligned} \tau_N &:= N, \\ \tau_n &:= \begin{cases} n, & \text{if } g(X_n) \geq \mathbb{E}[\alpha\, U_{n+1} \,|\, X_n], \\ \tau_{n+1}, & \text{otherwise.} \end{cases} \end{aligned} \tag{3}$$

In particular, at maturity N, the holder receives the final payoff, and the value of the option U_N is equal to the payoff $g(X_N)$. At each time prior to the maturity, the holder decides whether to exercise or not, depending on whether the current payoff

$g(X_n)$ is greater than the continuation value $c_n(X_n) := \mathbb{E}[\alpha\, U_{n+1} \mid X_n]$. Combining (1), (2) and (3), we can write the price at the initial time as

$$U_0 = \max\left(g(X_0), \mathbb{E}[\alpha^{\tau_1} g(X_{\tau_1})]\right).$$

In the following we approximate the price U_0 and continuation values $c_n(X_n)$ which are defined theoretically but cannot be computed directly.

2.3. **Monte Carlo simulation and backward recursion.** We assume having access to a procedure to sample discrete paths of X under $\mathbb{Q}$. A standard example is that X follows a certain stochastic differential equation (SDE) with known parameters. Therefore, we can sample m realizations of the stock price paths, where the i-th realization is denoted by $x_0, x_1^i, x_2^i, \dots, x_N^i$, with the fixed initial value x_0. For each realization, the cash flow realized by the holder when following the stopping strategy (3) is given by the backward recursion

$$p_N^i := g(x_N^i),$$

$$p_n^i := \begin{cases} g(x_n^i), & \text{if } g(x_n^i) \geq c_n(x_n^i), \\ \alpha p_{n+1}^i, & \text{otherwise.} \end{cases}$$

As p_1^i are samples of $\alpha^{\tau_1 - 1} g(X_{\tau_1})$, we have by the strong law of large numbers that almost surely

$$U_0 = \max\left(g(X_0), \lim_{m\to\infty} \frac{1}{m}\sum_{i=1}^{m} \alpha p_1^i\right). \tag{4}$$

2.4. **Randomized neural network approximation of the continuation value.** For each path i in $\{1, 2, \dots, m\}$ and each date n in $\{1, 2, \dots, N-1\}$, the continuation value is $c_n(x_n^i) = \mathbb{E}[\alpha U_{n+1} | X_n = x_n^i]$, where $c_n : \mathbb{R}^d \to \mathbb{R}$ describes the expected value of the discounted price αU_{n+1} if we keep the option until the next exercising date $n+1$, knowing the current values of the stocks X_n. We approximate this continuation value function by a neural network, where only the parameters of the last layer are learned. We refer to such a network as a randomized neural network. Even though the architecture of the neural network can be general, we present our algorithm with a simple dense shallow neural network, where the extension to deep networks is immediate. We call $\boldsymbol{\sigma} : \mathbb{R} \to \mathbb{R}$ the activation function. A common choice is $\boldsymbol{\sigma}(x) = \tanh(x)$, but there are many other suitable alternatives. For $K \in \mathbb{N}$, we define $\sigma : \mathbb{R}^{K-1} \to \mathbb{R}^{K-1}$, $\sigma(x) = (\boldsymbol{\sigma}(x_1), \dots, \boldsymbol{\sigma}(x_{K-1}))^\top$ for $x \in \mathbb{R}^{K-1}$. Let $\vartheta := (A, b) \in \mathbb{R}^{(K-1)\times d} \times \mathbb{R}^{K-1}$ be the parameters of the hidden layer which are randomly and identically sampled and not optimized. In general, A and b can be sampled from different distributions that are continuous and have support $\mathbb{R}$. The distributions and their parameters are hyperparameters of the randomized neural network that can be tuned. For simplicity we use a standard Gaussian distribution. Let us define

$$\phi : \mathbb{R}^d \to \mathbb{R}^K, x \mapsto \phi(x) = (\sigma(Ax + b)^\top, 1)^\top.$$

Let $\theta_n := ((A_n)^\top, b_n)^\top \in \mathbb{R}^{K-1} \times \mathbb{R}$ be the parameters that are optimized. Then, for each n the continuation value is approximated by

$$c_{\theta_n}(x) := \theta_n^\top \phi(x) = A_n^\top \sigma(Ax + b) + b_n.$$

2.5. **Least squares optimization of last layer's parameters $\boldsymbol{\theta}_n$.** While the parameters ϑ of the hidden layer are set randomly, the parameters θ_n of the last layer are found by minimizing the squared error of the difference between conditional expectation of the discounted future price and the approximation function. This is equivalent to finding θ_n which minimizes $\mathbb{E}[(c_{\theta_n}(x_n^i, n) - \alpha U_{n+1})^2 | X_n = x_n^i]$ for each time n in $\{1, 2, \dots, N-1\}$. The backward recursive Monte Carlo approximation of this expectation at time n yields the loss function

$$\psi_n(\theta_n) := \sum_{i=1}^{m} \left(c_{\theta_n}(x_n^i) - \alpha p_{n+1}^i \right)^2 . \tag{5}$$

As the approximation function c_{θ_n} is linear in the parameters θ_n, the minimizer can be found by ordinary least squares. It is given by the following closed form expression, which is well defined under the standard assumptions (see Theorems A.4 and A.5)

$$\theta_n = \alpha \left(\sum_{i=1}^{m} \phi(x_n^i)\phi^\top(x_n^i) \right)^{-1} \cdot \left(\sum_{i=1}^{m} \phi(x_n^i) p_{n+1}^i \right) .$$

2.6. **Splitting the data set into training and evaluation set.** The parameters θ_n are determined using 50% of the sampled paths (training data). Given θ_n, the remaining 50% of the sampled paths (evaluation data) are used to compute the option price. By definition, the continuation value c_n is a conditional expectation, which is not allowed to depend on the future values X_{n+k} for $0 < k \leq N - n$. On the training set, this might not be satisfied, since the loss function (5) uses the future values X_{n+k}. In particular, the neural network can suffer from overfitting to the training data, by memorizing the paths, instead of learning the continuation value. This is related to the maximization bias discussed in [61, Section 6.7]. By splitting the data into independent training and evaluation sets, we can however ensure that c_n evaluated on the evaluation set is independent of future values X_{n+k} of the evaluation set.

2.7. **Algorithm.** We first sample $2m$ paths and then proceed backward as follows. At maturity, the pathwise option price approximation is equal to the payoff, which means that $p_N^i = g(x_N^i)$. For each time n in $\{N-1, N-2, \dots, 0\}$, we first determine θ_n as described before using the paths $\{1, 2, \dots, m\}$. For all paths $i \in \{1, 2, \dots, 2m\}$ we then compare the exercise value $g(x_n^i)$ to the continuation value $c_{\theta_n}(x_n^i)$ and determine the path-wise option price approximation at time n as

$$p_n^i = \underbrace{g(x_n^i)}_{\text{payoff}} \underbrace{\mathbf{1}_{\{g(x_n^i) \geq c_\theta(x_n^i)\}}}_{\text{exercise}} + \underbrace{\alpha p_{n+1}^i}_{\text{discounted future price}} \underbrace{\mathbf{1}_{\{g(x_n^i) < c_\theta(x_n^i)\}}}_{\text{continue}} .$$

Finally, the second half of the paths $\{m+1, \dots, 2m\}$ is used to compute the option price approximation $p_0 = \max(g(x_0), \frac{1}{m}\sum_{i=m+1}^{2m} \alpha p_1^i)$. We call this algorithm, which is presented in Algorithm 1, randomized least squares Monte Carlo (RLSM).

2.8. **Guarantees of convergence.** We present results that guarantee convergence of the price computed with our algorithm to the correct price of the discretized American option. The formal results with precise definitions and proofs are given in Appendix A. In contrast to comparable results for neural networks [47, 10, 11], our results do not need the assumption that the optimal weights are found by some

Algorithm 1 Optimal stopping via randomized least squares Monte Carlo (RLSM)

```
Input: discount factor $\alpha$, initial value $x_0$
Output: price $p_0$
1: sample a random matrix $A \in \mathbb{R}^{(K-1)\times d}$ and a random vector $b \in \mathbb{R}^{K-1}$
2: simulate $2m$ paths of the underlying process $(x_1^i, \dots, x_N^i)$ for $i \in \{1, \dots, 2m\}$
3: for each path $i \in \{1, \dots, 2m\}$, set $p_N^i = g(x_N^i)$
4: for each time $n \in \{N-1, \dots, 1\}$
  a: for each path $i \in \{1, \dots, 2m\}$, set $\phi(x_n^i) = (\sigma(Ax_n^i + b)^\top, 1)^\top \in \mathbb{R}^K$
  b: set $\theta_n = \alpha \left(\sum_{i=1}^m \phi(x_n^i)\phi^\top(x_n^i)\right)^{-1} \left(\sum_{i=1}^m \phi(x_n^i) p_{n+1}^i\right)$
  c: for each path $i \in \{1, \dots, 2m\}$
       set $p_n^i = g(x_n^i)\mathbf{1}_{g(x_n^i)\geq\theta_n^\top\phi(x_n^i)} + \alpha p_{n+1}^i \mathbf{1}_{g(x_n^i)<\theta_n^\top\phi(x_n^i)}$
5: set $p_0 = \max(g(x_0), \frac{1}{m}\sum_{i=m+1}^{2m} \alpha p_1^i)$
```

optimization scheme like stochastic gradient descent. Instead, our algorithms imply that the optimal weights are found and used.

Theorem 2.1 (informal)**.** *As the number of sampled paths m and the number of random basis functions K go to ∞, the price p_0 computed with Algorithm 1 converges to the correct price of the Bermudan option.*

2.9. **Possible extensions.** When the set of pricing measures $\mathcal{Q}$ has more than one element (in case of an incomplete market), the option price is given by $\sup_{\mathbb{Q}\in\mathcal{Q}} U_0^{\mathbb{Q}}$, where $U^{\mathbb{Q}}$ is defined as in (1). Assuming that we can sample from a finite subset $\mathcal{Q}_1 \subset \mathcal{Q}$, this price can be approximated by first computing the price for each measure in $\mathcal{Q}_1$ and then taking the maximum of them.

For simplicity we assume that the payoff function only takes the current price as input. However, all our methods and results stay valid if $g(X_n)$ is replaced by a square integrable $\mathcal{F}_n$-measurable random variable Z_n, where $\mathcal{F}_n$ denotes the information available up to time n. In the case that $(Z_n)_{1\leq n\leq N}$ is Markov, Algorithm 1 and Algorithm 2 (Section 3) can be used; otherwise, Algorithm 3 (Section 4) has to be used, to deal with the path dependence. In the following sections we stick to the notation $g(X_n)$ for the payoff, keeping in mind that the extension to a general Z_n is also valid there.

Similarly, a more general discounting can be incorporated, by assuming that Z_n is given in discounted terms and setting $\alpha = 1$.

3. Optimal stopping via randomized reinforcement learning.

In order to avoid approximating the continuation value at each single date $n \in \{1, \dots, N-1\}$ with a different function, as is done in Section 2, we can directly learn the continuation function which also takes the time as argument. Hence, instead of having a different function $c_{\theta_n}(x_n^i)$ for each date n, we learn one function which is used for all dates n. As previously, we define the parameters of the hidden layer $\vartheta := (A, b) \in \mathbb{R}^{(K-1)\times(d+2)} \times \mathbb{R}^{K-1}$, which are randomly chosen and not optimized, and $\phi : \mathbb{R}^{d+2} \to \mathbb{R}^K$, $\phi(n, x) = (\sigma(A\tilde{x}_n + b)^\top, 1)^\top$, where $\tilde{x}_n = (n, N-n, x_n^\top)^\top$. Let $\theta \in \mathbb{R}^K$ define the parameters that are optimized, and then the continuation value is approximated by

$$c_\theta(n, x) := \theta^\top \phi(n, x)\,.$$

Instead of having a loop backward in time with N steps, we iteratively improve the approximation c_θ. More precisely, we start with some (random) initial weight θ_0 and then iteratively improve it by minimizing the difference between the continuation function c_{θ_ℓ} and the prices p computed with the previous weight $\theta_{\ell-1}$. Moreover, differently than in Section 2, we use the continuation value for the decision whether to continue *and* for the approximation of the discounted future price, as in [64]. This second algorithm can be interpreted as a randomized fitted Q-iteration (RFQI) and is presented in Algorithm 2. It is a very simple type of reinforcement learning, where the agent has only two possible actions, and the agent's decision does not influence the transitions of the state. In particular the agent's decision does not influence the evolution of the underlying stocks. As a reinforcement learning method, it is based on the assumption that the optimization problem can be modeled by a Markov decision process.

Algorithm 2 Optimal stopping via randomized fitted Q-Iteration (RFQI)

```
Input: discount factor $\alpha$, initial value $x_0$
Output: price $p_0$
1: sample a random matrix $A \in \mathbb{R}^{(K-1)\times(d+2)}$ and a random vector $b \in \mathbb{R}^{K-1}$
2: simulate $2m$ paths of the underlying process $(x_1^i, \dots, x_N^i)$ for $i \in \{1, \dots, 2m\}$
3: initialize weights $\theta_0 = 0 \in \mathbb{R}^K$ and set $\ell = 0$
4: until convergence of $\theta_\ell$
  a: for each path $i \in \{1, \dots, 2m\}$
    i: set $p_N^i = g(x_N^i)$
    ii: for each date $n \in \{1, \dots, N-1\}$
        set $\phi(n, x_n^i) = (\sigma(A\tilde{x}_n^i + b), 1) \in \mathbb{R}^K$
        set $p_n^i = \max(g(x_n^i), \phi(n, x_n^i)^\top \theta_\ell)$
  b: set
     $$\theta_{\ell+1} = \alpha \left( \sum_{n=1}^{N} \sum_{i=1}^{m} \phi(n, x_n^i)\phi^\top(n, x_n^i) \right)^{-1} \left( \sum_{n=1}^{N} \sum_{i=1}^{m} \phi(n, x_n^i) p_{n+1}^i \right) \in \mathbb{R}^K$$
  c: set $\ell \leftarrow \ell + 1$
5: set $p_0 = \max(g(x_0), \frac{1}{m} \sum_{i=m+1}^{2m} \alpha p_1^i)$
```

3.1. Guarantees of convergence.

We present results that guarantee convergence of the price computed with our algorithm to the correct price of the discretized American option. The formal results with precise definitions and proofs are given in Appendix B.

Theorem 3.1 (informal)**.** *As the number of iterations L, the number of sampled paths m and the number of random basis functions K go to ∞, the price p_0 computed with Algorithm 2 converges to the correct price of the Bermudan option.*

4. Optimal stopping via randomized recurrent neural networks for non-Markovian processes.

For non-Markovian processes, for each date n, the continuation function is no longer a function of the last stock price, $c_n(X_n)$, but a function depending on the entire history $c_n(X_0, X_1, \dots, X_{n-1}, X_n)$. More precisely, the continuation value is now defined by $c_n := \mathbb{E}[\alpha g(X_{n+1}) \,|\, \mathcal{F}_n]$ where $\mathcal{F}_n$ denotes the information available up to time n. Therefore, we replace the randomized feed-forward neural network by a randomized recurrent neural network (randomized RNN), which can utilize the entire information of the path up to the current time

$(x_0, x_1, \dots, x_{n-1}, x_n)$. In particular, we define the parameters of the hidden layer $\vartheta := (A_x, A_h, b) \in \mathbb{R}^{(K-1)\times d} \times \mathbb{R}^{(K-1)\times(K-1)} \times \mathbb{R}^{K-1}$, which are randomly sampled and not optimized. Their distributions and parameters, which do not have to be the same for A_x and A_h, are hyperparameters that can be tuned. These tuning parameters are more important in this case, as they determine the interplay between past and new information. Moreover, we define

$$\phi : \mathbb{R}^d \times \mathbb{R}^K \to \mathbb{R}^{K+1}, \quad (x,h) \mapsto \phi(x,h) = (\sigma(A_x x + A_h h + b)^\top, 1)^\top$$

and $\theta_n := ((A_n)^\top, b_n)^\top \in \mathbb{R}^{K-1} \times \mathbb{R}$, the parameters that are optimized. Then, for each n, the continuation value is recursively approximated by

$$\begin{cases} h_n & := \sigma(A_x x_n + A_h h_{n-1} + b), \\ c_{\theta_n}(h_n) & := A_n^\top h_n + b_n = \theta_n^\top \phi(x_n, h_{n-1}), \end{cases} \tag{6}$$

with $h_{-1} := 0$. We call this algorithm, which is presented in Algorithm 3, randomized recurrent least squares Monte Carlo (RRLSM).

Algorithm 3 Optimal stopping via randomized recurrent least squares Monte Carlo (RRLSM)

```
Input: discount factor $\alpha$, initial value $x_0$, initial latent variable $h_{-1} = 0$
Output: price $p_0$
1: sample random matrices $A_x \in \mathbb{R}^{(K-1)\times d}$, $A_h \in \mathbb{R}^{(K-1)\times(K-1)}$ and a random vector $b \in \mathbb{R}^{K-1}$
2: simulate $2m$ paths of the underlying process $(x_1^i, \dots, x_N^i)$ for $i \in \{1, \dots, 2m\}$
3: for each path $i \in \{1, \dots, 2m\}$, set $p_N^i = g(x_N^i)$
4: for each date $n \in \{0, \dots, N-1\}$
  a: for each path $i \in \{1, \dots, 2m\}$, set $h_n^i = \sigma(A_x x_n^i + A_h h_{n-1}^i + b)$
5: for each date $n \in \{N-1, \dots, 1\}$
  a: for each path $i \in \{1, \dots, 2m\}$, set $\phi_n^i = ((h_n^i)^\top, 1)^\top \in \mathbb{R}^K$
  b: set $\theta_n = \alpha \left(\sum_{i=1}^m \phi_n^i (\phi_n^i)^\top\right)^{-1} \left(\sum_{i=1}^m \phi_n^i p_{n+1}^i\right)$
  c: for each path $i \in \{1, \dots, 2m\}$
       set $p_n^i = g(x_n^i)\mathbf{1}_{g(x_n^i) \geq \theta_n^\top \phi_n^i} + \alpha p_{n+1}^i \mathbf{1}_{g(x_n^i) < \theta_n^\top \phi_n^i}$
6: set $p_0 = \max(g(x_0), \frac{1}{m}\sum_{i=m+1}^{2m} \alpha p_1^i)$
```

4.1. **Guarantees of convergence.**

We present results that guarantee convergence of the price computed with our algorithm to the correct price of the discretized American option. The formal results with precise definitions and proofs are given in Appendix C.

Theorem 4.1 (informal)**.** *As the number of sampled paths m and the number of random basis functions K go to ∞, the price p_0 computed with Algorithm 3 converges to the correct price of the Bermudan option.*

5. **Upper bounds for American option prices.**

So far we have approximated the value of an American option by computing the optimal stopping time (3) with which the value of the Snell envelope (2) can be determined. Since our algorithms only *approximate* the optimal stopping time of this maximization problem, the resulting price is a lower bound for the true value. The advantage of this method is that it not only provides an approximation for the price but also provides a decision rule for when to stop.

An upper bound of the true value is naturally implied by the dual method introduced by [55]. In particular, [55, Theorem 2.1] yields that the starting value of the Snell envelope (2), which is the price of the American option, can equivalently be written as the minimization problem

$$U_0 = \inf_{M \in \mathcal{M}_0} \mathbb{E}\left[\sup_{0 \le n \le N} (Z_n - M_n) \right], \tag{7}$$

where Z_n denotes the *discounted* payoff process (see Section 2.9), and $\mathcal{M}_0$ is the set of $(\mathcal{F}_n)$-martingales starting at 0. As explained in [10, Section 3.2], the minimizer of (7) is given by the martingale part of the Doob-Meyer decomposition of the Snell envelope $(\tilde{U}_n)_{0 \le n \le N}$ for the discounted payoff process. This martingale is defined through

$$\begin{aligned} M_0^U &:= 0, \\ M_n^U - M_{n-1}^U &:= \tilde{U}_n - \mathbb{E}[\tilde{U}_n | \mathcal{F}_{n-1}] = \max(Z_n, \mathbb{E}[\tilde{U}_{n+1} | \mathcal{F}_n]) - \mathbb{E}[\tilde{U}_n | \mathcal{F}_{n-1}], \end{aligned} \tag{8}$$

where we used (the discounted version of) (1) for the last equality. Since our algorithms compute the continuation values $c_n = \mathbb{E}[\tilde{U}_{n+1} | \mathcal{F}_n]$ (written in the most general form; see Section 4), we can compute an upper bound approximation of the option price together with the lower bound nearly without additional costs[2] via (7) and (8) as

$$U_0 = \mathbb{E}\left[\sup_{0 \le n \le N} (Z_n - M_n^U) \right],$$

with

$$M_0^U = 0, \quad M_n^U - M_{n-1}^U = \max(Z_n, c_n) - c_{n-1}.$$

Moreover, given our methods to compute the lower and upper bound of an American option price, confidence intervals can be computed exactly as described in [10, Section 3.3].

Algorithm 4 is the extension of Algorithm 1, where the upper bound for the option price is computed in addition to the lower bound. The extensions of Algorithms 2 and 3 work similarly.

6. Related work.

We present the most relevant approaches for the optimal stopping problem: backward induction either with basis functions or with neural networks and reinforcement learning. Moreover, we explain the connection of our algorithms to randomized neural networks and reservoir computing techniques.

[2]We only need the additional computation and storage of the martingale differences $M_n^U - M_{n-1}^U$, the computation of M^U as its cumulative sum and finally the computation of the maximum over $Z_n - M_n^U$. Since no additional loop or simulation is needed, the computational costs stay nearly the same. In particular, this may be considered as an advantage over the algorithm presented by [10], where the computations of the lower bound as well as additional time consuming computations (approximately doubling the total computation time) are needed to get an approximation of the upper bound.

Algorithm 4 Upper and lower bounds for Bermudan option price via randomized least squares Monte Carlo (RLSM)

```
Input: discount factor $\alpha$, initial value $x_0$
Output: lower price bound $p_0^l$, upper price bound $p_0^u$
1: sample a random matrix $A \in \mathbb{R}^{(K-1)\times d}$ and a random vector $b \in \mathbb{R}^{K-1}$
2: simulate $2m$ paths of the underlying process $(x_1^i, \dots, x_N^i)$ for $i \in \{1, \dots, 2m\}$
   and compute the discounted payoffs $z_n^i = g(x_n^i) * \alpha^n$ for $1 \le i \le 2m$, $1 \le n \le N$
3: for each path $i \in \{1, \dots, 2m\}$, set $p_N^i = z_N^i$ and $c_N^i = 0$
4: for each time $n \in \{N-1, \dots, 0\}$
  a: for each path $i \in \{1, \dots, 2m\}$, set $\phi(x_n^i) = (\sigma(Ax_n^i + b)^\top, 1)^\top \in \mathbb{R}^K$
  b: set $\theta_n = \left(\sum_{i=1}^m \phi(x_n^i)\phi^\top(x_n^i)\right)^{-1} \left(\sum_{i=1}^m \phi(x_n^i) p_{n+1}^i\right)$
  c: for each path $i \in \{1, \dots, 2m\}$
    i: set $c_n^i = \theta_n^\top \phi(x_n^i)$
    ii: set $p_n^i = z_n^i \mathbf{1}_{z_n^i \ge c_n^i} + p_{n+1}^i \mathbf{1}_{z_n^i < c_n^i}$
    iii: set $\Delta M_{n+1}^i = \max(z_{n+1}^i, c_{n+1}^i) - c_n^i$
5: set $p_0^l = \max(g(x_0), \frac{1}{m}\sum_{i=m+1}^{2m} p_1^i)$
6: set $p_0^u = \frac{1}{m}\sum_{i=m+1}^{2m} \left(\max_{0\le n\le N}(z_n^i - \sum_{k=1}^n \Delta M_k^i)\right)$
```

6.1. **Optimal stopping.**

Numerous works studied the optimal stopping problem via different approaches. A common approach consists of using a regression based method to estimate the continuation value [62, 7, 17, 63, 64, 51, 59, 14, 15, 46, 24, 43], or the optimal stopping boundary [54, 2, 31]. A different approach uses quantization [4, 5]. A dual approach was developed and extended in [55, 38, 56]. [6] studied Lenglart's theory of Meyer-sigma-fields and El Karoui's theory of optimal stopping [28]. An in depth review of the different methods is given in [13, 53].

6.1.1. *Optimal stopping via backward induction.* Among the most popular approaches are the backward induction methods introduced by [64] and [51]. [64] uses the approximated continuation value to estimate the current price, by using the backward recursion

$$p_n^i = \max(g(x_n^i), c_{\theta_n}(x_n^i)) \,. \tag{9}$$

Instead, [51] uses the continuation value only for the decision to stop or to continue, yielding

$$p_n^i = \begin{cases} g(x_n^i), & \text{if } g(x_n^i) \ge c_{\theta_n}(x_n^i) \\ \alpha p_{n+1}^i, & \text{otherwise.} \end{cases} \tag{10}$$

The second algorithm is more robust, as the approximation is only used for the decision and not for the estimation of the price. Hence, the method proposed by [51] is the most used method in the financial industry and can be considered state-of-the-art. In both papers, the approximation $c_\theta(x_n^i) = \theta^\top \phi(x_n^i)$ is used, where $\phi = (\phi_1, \dots, \phi_K)$ is a set of K basis functions, and $\theta \in \mathbb{R}^K$ are the trainable weights. Possible choices for the basis functions proposed in [51] are Laguerre, Hermite, Legendre, Chebyshev, Gegenbauer and Jacobi polynomials. While they have the advantage of having convergence guarantees, both algorithms do not easily scale to high dimensional problems since the number of basis functions usually grows polynomially or even exponentially [51, Section 2.2] in the number of stocks. One direction of research to overcome this problem is to apply dimension reduction techniques [8].

6.1.2. *Optimal stopping via backward induction using neural networks.* Another idea to overcome this issue was proposed by [45], which consists of approximating the continuation value by a neural network

$$f_\theta(x_n^i) \approx c_\theta(x_n^i).$$

That way, the features are learned contrary to the basis functions which must be chosen. While [45] used the backward recursion (9) introduced by [64], both [47] and [11] used the backward recursion (10) suggested by [51]. Instead of approximating the continuation value, [10] suggested approximating the whole indicator function present in (10) by a neural network $f_{\theta_n}(x_n^i) \approx \mathbf{1}_{\{g(x_n^i) \geq c(x_n^i)\}}$. Then, the current price is estimated by

$$p_n^i \quad = \quad g(x_n^i) \underbrace{f_{\theta_n}(x_n^i)}_{\text{stop}} + \alpha p_{n+1}^i \underbrace{\left(1 - f_{\theta_k}(x_n^i)\right)}_{\text{continue}} .$$

Moreover, instead of minimizing the loss function (5) in order to find a good approximation of the continuation function, [10] optimized the parameters by directly maximizing the option price $\psi_n(\theta_n) = \frac{1}{m}\sum_{i=1}^m \alpha p_n^i$.

All those algorithms use stochastic gradient methods to determine the parameters of the neural networks. They have to find the parameters of $N-1$ neural networks (using a different neural network for each date). Since they use stochastic gradient methods with a non-convex loss function they cannot provide theoretical convergence guarantees, without the strong assumption that they actually find the optimal parameters.

6.1.3. *Optimal stopping via reinforcement learning.* By its nature, reinforcement learning is closely related to the dynamic programming principle as shown in [61, 12]. Moreover, the optimal stopping problem is well studied as an application of reinforcement learning [63, 64, 66, 49]. In all those methods, a linear approximator is used (linear combination of basis functions), similarly to the LSM method [51]. If a standard set of basis functions that grows polynomially in the dimension is used, then these methods suffer from the curse of dimensionality. In particular, they cannot practically be scaled to high dimensions, as can be seen in our numerical results. To the best of our knowledge, our approach constitutes the first time that randomized neural networks are used to approximate the value function in reinforcement learning.

6.2. **Randomized neural networks and reservoir computing.** In RLSM and RFQI we use a neural network with randomly sampled and fixed hidden layers, where only the last layer is reinitialized and trained at each time $n \in \{N-1, \dots, 1\}$. The architecture used at each time can be interpreted as a neural network with random weights (NNRW) studied and reviewed in [16], where a universality result was provided in [42]. Randomized neural networks as approximation functions were also studied by [36].

Randomized recurrent neural networks are an extension of randomized neural networks. A recurrent neural network (RNN) where the parameters are randomly generated and fixed and only the readout map is trained is known as a reservoir. Reservoir computing not only reduces the computation time but also outperforms classical, fully trained RNNs in many tasks [58, 65, 52, 30]. Similarly, as in reservoir computing, in our randomized recurrent neural network algorithm RRLSM, the parameters of the hidden layers are randomly sampled and fixed thereafter.

However, while reservoir computing trains only one readout map which has the same parameters for all times, we train a different readout map for each single time $n \in \{N-1, \dots, 1\}$, similarly to RLSM.

6.3. **Backward induction versus reinforcement learning.** Backward induction is an (approximate) dynamic programming (ADP) approach. While [61] regards ADP as a class of RL algorithms, we distinguish these two approaches in this work, because of their different algorithmic structure and their different ways of using the training data. In particular, backward recursion computes the approximation of the continuation value for each date sequentially. More precisely, it starts at the final date and goes backward in time. For the approximation at each date, only the data of this date is used. In contrast to this, RL starts with an initial approximation that is applied for all dates and iteratively improves this approximation. This way, the data of all dates is used to improve the approximation of all dates. In comparison to backward recursion, this can be interpreted as a type of transfer learning between the dates.

7. Experiments.

There are numerous ways to empirically evaluate optimal stopping approaches. We choose the most studied settings that were considered in the American option pricing literature. In particular, we only consider synthetic data. Applications to real data involve model calibration, which is an independent problem and finally results in applying the optimal stopping algorithm to synthetically generated data again.

Besides our algorithms, we also implemented the baselines and provided all of them at https://github.com/HeKrRuTe/OptStopRandNN.

7.1. **Experimental setup.** In our experiments, we mainly focus on the computation and comparison of lower bound approximations of American option prices. However, in Section 7.5 we present experiments where also upper bound approximations are computed with RLSM based on the derivation in Section 5.

The evaluation of all the algorithms was done on the same computer, a dedicated machine with 2×Intel Xeon CPU E5-2697 v2 (12 Cores) 2.70GHz and 256 GiB of RAM.

7.1.1. *Baselines (LSM, NLSM, DOS and FQI).* We compare RLSM and RFQI to three backward induction algorithms and one reinforcement learning approach: first, the state-of-the-art least squares Monte Carlo (LSM) [51]; second, the algorithm proposed by [47], where the basis functions are replaced by a deep neural network (NLSM); third, deep optimal stopping (DOS) [10], where instead of the continuation value, the whole indicator function of the stopping decision is approximated by a neural network; and finally, the fitted Q-iteration (FQI) presented as the second algorithm in [63]. [49] studied and compared two reinforcement learning based methods (FQI and LSPI) to solve the optimal stopping problem. Since FQI always worked better in our experiments, we only show comparisons to this algorithm. Our aim is to compare the main concepts of all the different algorithms in a fair way, so we leave away certain (more sophisticated) particularities unique to each of them.

7.1.2. *Choice of basis functions for the baselines.* There are many possible choices for the set of basis functions. [51] proposed using the first three weighted Laguerre polynomials for LSM, and [49] added three additional basis functions of the date for FQI. While the size of this set of basis functions scales linearly with the dimension, it does not include any interaction terms. The classical polynomial basis functions up to the second order are the easiest way to include coupling terms in the basis. To deal with the time dependence of FQI, the relative date t/T and $1 - t/T$ are added as additional coordinates to the d-dimensional stock vector. The size of this basis grows quadratically in the dimension d, i.e., it has $1 + 2d + d(d-1)/2$ elements for LSM and for FQI d is replaced by $d+2$. The results obtained with the classical polynomials up to degree two were better than with the weighted Laguerre polynomials for LSM and FQI, so we only present these results in our tables. For large d the computations of LSM and FQI did not terminate within a reasonable amount of time (several hours) and therefore were aborted.

7.1.3. *No regularization for LSM and FQI.* While drastically increasing the number of hidden nodes without increasing the number of paths or applying penalization led to overfitting for RLSM and RFQI, this was not observed for LSM and FQI. In particular, for LSM, Ridge regression (L^2-penalisation) was tested without leading to better results than standard linear regression. Moreover, comparing the results of FQI, RFQI and DOS for growing dimensions shows that overfitting does not become a problem when more basis functions are used. Therefore, also for FQI standard linear regression was used as suggested by [63].

7.1.4. *Architecture of neural networks.* In order to have a fair comparison in terms of accuracy and in terms of computation time, we use the same number of hidden layers and nodes per layer for all the algorithms.

- We observed that one hidden layer was sufficient to have a good accuracy (an increase of the number of the hidden layers did not lead to better accuracy). Therefore, NLSM, DOS and all algorithms that we proposed have only one hidden layer.
- We use 20 nodes for the hidden layer. Importantly, we do not claim that this choice is optimal for any of the methods. For RFQI the number of nodes is set to the minimum between 20 and the number of stocks, for stability reasons.
- Leaky ReLU is used for RLSM and RFQI, and tanh is used for the randomized recurrent neural network RRLSM. For NLSM and DOS, we use the suggested activation functions, Leaky ReLU for NLSM and ReLU and sigmoid for DOS.
- The parameters (A, b) of the random neural networks of RLSM and RFQI are sampled using a standard normal distribution with mean 0 and standard deviation 1. Different hyper-parameters were tested, but they did not have a big influence on the results so we kept the standard choice.
- For the randomized recurrent neural network of RRLSM, 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. The same holds for tested path-dependent versions of RFQI; however, none of the hyper-parameters performed very well as shown below.
- Some of the reference methods suggest to use the payoff as additional input, while others do not or leave this open. Therefore, we tested using the payoff as input and not using it for each method in each experiment. We came to the conclusion that the backward induction algorithms (LSM, DOS, NLSM, RLSM) usually work slightly better with, while the reinforcement learning algorithms (FQI, RFQI) usually work slightly better without the payoff. Hence, we show these results.

- As suggested by the authors, we used batch normalization for the implementation of DOS.

7.2. **The Markovian case – Bermudan option pricing.** First, we evaluate RLSM and RFQI in the standard Markovian setting of Bermudan option pricing with different stock price models and payoff functions.

7.2.1. *Stock models (Black–Scholes and Heston).* We test our algorithm on two multidimensional stochastic models, Black–Scholes and Heston with fixed parameters and standard discounting factor $\alpha = e^{-rT/N}$. For each model we sample $m = 20,000$ paths on the time interval $[0, 1]$, i.e., with maturity $T = 1$, using the Euler-scheme with $N = 10$ equidistant dates. As explained in Section 2.6, we use half of the paths as the training set and the second half to compute the approximated price using the trained continuation value function (or decision function in the case of DOS).

The Black–Scholes model for a max call option is a widely used example in the literature [51, 47, 10]. The stochastic differential equation (SDE) describing this model is

$$dX_t = (r - \delta)X_t dt + \sigma X_t dW_t,$$

with $X_0 = x_0$, where $(W_t)_{t\geq 0}$ is a d-dimensional Brownian motion. If not stated differently, we choose the rate $r = 0\%$, the dividend rate $\delta = 0\%$, the volatility $\sigma = 20\%$ and the initial stock price $x_0 \in \{80, 100, 120\}$.

To increase the complexity, we also compare the algorithms on the Heston model [40], which is also used in [47]. The SDE describing this model is

$$\begin{aligned} dX_t &= (r - \delta)X_t dt + \sqrt{v_t} X_t dW_t, \\ dv_t &= -\kappa(v_t - v_\infty)dt + \sigma\sqrt{v_t} dB_t, \end{aligned} \tag{11}$$

with $X_0 = x_0$ and $v_0 = \nu_0$, where $(W_t)_{t\geq 0}$ and $(B_t)_{t\geq 0}$ are two d-dimensional Brownian motions correlated with coefficient $\rho \in (-1, 1)$. Here, X is the stock price, and v is the stochastic variance process. If not stated differently, we choose the drift $r = 0\%$, the dividend rate $\delta = 0\%$, the volatility of volatility $\sigma = 20\%$, the long term variance $v_\infty = 0.01$, the mean reversion speed $\kappa = 2$, the correlation $\rho = -30\%$, the initial stock price $x_0 = 100$ and the initial variance $\nu_0 = 0.01$ (in particular, the Feller condition $2\kappa v_\infty > \sigma^2$ is not satisfied, so v_t might touch the value 0 but is reflected immediately). Since the Heston model is Markovian only if the price and the variance (X_t, v_t) are observed simultaneously, we give both values as inputs to the algorithms here, and denote this below by "Heston (with variance)".

7.2.2. *Payoffs (max call, geometric put, basket call and min put).* We test our algorithms on four different types of options. First, we consider the max call option as it is a classical example used in optimal stopping [47, 10]. The payoff of a max call option is defined by $g(x) = (\max(x_1, x_2, \ldots, x_d) - K)_+$ for any $x = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. Moreover, we consider the geometric put option, used in [47], with payoff $g(x) = (K - (\prod_{i=1}^d x_i)^{1/d})_+$. We also test our approach on a basket call option [37], where the payoff is given by $g(x) = (\frac{1}{d}\sum_{i=1}^d x_i - K)_+$, and a min put option with payoff $g(x) = (K - \min(x_1, x_2, \ldots, x_d))_+$. For all these payoffs, the strike K is set to 100, unless stated differently.

7.2.3. *Reference prices.* In some cases reference prices can be computed as additional baselines. All call options where the underlying stocks have a rate $r \geq 0$ and dividend $\delta = 0$ are optimally executed at maturity. Therefore, the price of the American option and of the corresponding European option are the same under these constraints [29]. For all examples where this is the case, we compute the European option price (EOP) as an approximation of the correct American option price.

Moreover, as explained in [47], geometric put options on d-dimensional stocks following Black–Scholes are equivalent to one-dimensional put options on a 1-dimensional stock following Black–Scholes with adjusted parameters. The 1-dimensional problem can be priced efficiently with the CRR binomial-tree method (B) [21]. With the adjusted parameters $\hat{\sigma} = \frac{\sigma}{\sqrt{d}}$ and $\hat{\delta} = \delta + \frac{\sigma^2 - \hat{\sigma}^2}{2}$ the binomial-tree model is defined with factors for the stock price going up and down $\gamma_{up} = \exp(\hat{\sigma}\sqrt{T/N})$, $\gamma_{down} = \frac{1}{\gamma_{up}}$, probabilities to go up and down $p = \frac{\exp((r-\hat{\delta})T/N) - \gamma_{down}}{\gamma_{up} - \gamma_{down}}$, $1-p$ and step-wise discounting factor $\exp(-rT/N)$. The price computed with this method converges to the correct price under the Black–Scholes model as $N \to \infty$ [21]. Hence, this method yields good approximations of the correct option price for large N. Whenever applied, we use $N = 10,000$ for the binomial-tree method. While in the first case of call options, the optimal stopping problem has an easy solution, i.e., to wait until maturity, this is not the case here, where the optimal stopping problem is more complex.

The remaining options, i.e., call options with $\delta > 0$, put options with $r > 0$ and geometric put option with underlying stocks following a Heston model, also constitute more complex stopping problems, where no efficient methods to compute the (approximately) correct price are available. Therefore, we evaluate the performance of the algorithms by comparing the approximated prices directly. Since these prices are computed on unseen paths for all algorithms, where at each time, the algorithm can only decide whether to exercise or not, higher prices imply better performance of the algorithms.

7.2.4. *Results and discussion.* All algorithms are run 10 times in parallel, and the mean and standard deviation (in parenthesis) of the prices and the median of the corresponding computation times are reported. In particular, the computation times do not include the time for generating the stock paths, since the main interest is in the actual time the algorithms need to compute prices, and paths can be generated offline and stored. In the following discussion, we always compare computation times for large d, since random machine influences have less impact there.

In all cases RLSM and RFQI are the fastest algorithms while achieving at least similar prices to the best performing baselines. Their biggest strengths are high dimensional problems ($d \geq 500$), where this speed-up becomes substantial.

In these high dimensional problems, RLSM performs as good as or outperforms the baselines in terms of prices, even tough RLSM has much less trainable parameters than DOS and NLSM. Moreover, RFQI achieves the highest prices there and therefore works best, while having considerably less trainable parameters, since only one neural network (with a random hidden layer) of the respective size is used for all exercise dates. In particular, RFQI has only 21 trainable parameters, compared to more than $20dN$ for DOS and NLSM.

Table 1. Max call option on Black–Scholes for different numbers of stocks d and varying initial stock price x_0.

		price							duration						
d	x_0	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP
5	80	5.23 (0.07)	5.12 (0.12)	5.19 (0.09)	5.28 (0.12)	5.26 (0.10)	5.20 (0.06)	5.31 (0.05)	11s	9s	0s	0s	2s	0s	0s
	100	24.95 (0.14)	24.64 (0.21)	24.72 (0.15)	24.91 (0.16)	24.96 (0.17)	25.00 (0.19)	24.97 (0.15)	11s	8s	2s	0s	2s	0s	0s
	120	49.73 (0.21)	49.45 (0.18)	49.47 (0.22)	49.62 (0.25)	49.68 (0.22)	49.75 (0.17)	49.77 (0.15)	11s	7s	2s	0s	2s	0s	0s
10	80	9.20 (0.07)	9.19 (0.14)	8.82 (0.15)	9.24 (0.11)	9.25 (0.12)	9.25 (0.10)	9.27 (0.09)	28s	7s	1s	0s	6s	0s	0s
	100	34.33 (0.15)	34.03 (0.17)	33.69 (0.20)	34.28 (0.11)	34.25 (0.19)	34.17 (0.11)	34.26 (0.09)	29s	7s	2s	0s	7s	0s	0s
	120	60.94 (0.24)	60.90 (0.20)	60.33 (0.25)	61.08 (0.23)	61.10 (0.19)	61.07 (0.21)	61.20 (0.13)	29s	7s	2s	0s	6s	0s	0s
50	80	22.45 (0.11)	23.17 (0.10)	21.78 (0.34)	22.03 (0.16)	23.51 (0.13)	23.42 (0.11)	23.52 (0.09)	8m39s	8s	2s	0s	6m28s	1s	0s
	100	53.49 (0.10)	53.93 (0.12)	52.15 (0.60)	52.44 (0.21)	54.24 (0.09)	54.23 (0.08)	54.37 (0.09)	8m42s	8s	3s	0s	6m57s	1s	0s
	120	84.31 (0.12)	84.72 (0.12)	82.48 (0.79)	82.98 (0.16)	85.03 (0.18)	85.00 (0.20)	85.28 (0.07)	8m46s	9s	3s	0s	7m 4s	1s	0s
100	80	24.02 (0.21)	29.56 (0.13)	27.08 (0.47)	28.50 (0.06)	29.59 (0.15)	29.88 (0.08)	29.95 (0.08)	39m44s	13s	3s	0s	1h23m39s	1s	0s
	100	56.83 (0.18)	61.84 (0.26)	58.99 (0.62)	60.58 (0.10)	62.07 (0.15)	62.32 (0.16)	62.43 (0.08)	40m42s	13s	4s	0s	1h23m28s	1s	0s
	120	88.05 (0.31)	94.26 (0.16)	90.48 (0.89)	92.71 (0.08)	94.41 (0.17)	94.65 (0.14)	94.99 (0.14)	40m25s	13s	4s	0s	1h22m15s	1s	0s
500	80	-	42.54 (0.16)	39.45 (0.61)	43.00 (0.07)	-	44.15 (0.09)	44.34 (0.08)	-	53s	11s	1s	-	1s	0s
	100	-	78.27 (0.16)	74.23 (1.03)	78.80 (0.10)	-	80.21 (0.13)	80.45 (0.08)	-	53s	12s	1s	-	1s	0s
	120	-	113.83 (0.18)	108.60 (1.01)	114.54 (0.09)	-	116.26 (0.19)	116.48 (0.11)	-	53s	12s	1s	-	1s	0s
1000	80	-	47.91 (0.08)	45.37 (0.91)	49.13 (0.11)	-	50.14 (0.10)	50.32 (0.07)	-	1m34s	20s	2s	-	1s	0s
	100	-	84.99 (0.19)	81.06 (0.56)	86.40 (0.08)	-	87.70 (0.09)	87.93 (0.08)	-	1m35s	20s	3s	-	1s	0s
	120	-	121.98 (0.11)	118.61 (1.31)	123.68 (0.08)	-	125.19 (0.14)	125.48 (0.09)	-	1m36s	19s	2s	-	1s	0s
2000	80	-	53.14 (0.13)	51.45 (0.82)	55.13 (0.09)	-	56.03 (0.04)	56.27 (0.07)	-	2m57s	34s	5s	-	2s	0s
	100	-	91.43 (0.13)	89.84 (0.67)	93.87 (0.12)	-	95.00 (0.14)	95.31 (0.06)	-	3m 2s	39s	5s	-	2s	0s
	120	-	129.77 (0.15)	127.14 (1.30)	132.69 (0.15)	-	134.09 (0.08)	134.34 (0.10)	-	2m57s	37s	4s	-	2s	0s

Table 2. Max call option on Heston (with variance) for different numbers of stocks d.

	price							duration						
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP
5	8.34 (0.08)	8.36 (0.07)	8.22 (0.09)	8.37 (0.07)	8.25 (0.03)	8.33 (0.07)	8.23 (0.04)	31s	6s	3s	0s	8s	0s	0s
10	11.81 (0.06)	11.83 (0.07)	11.51 (0.12)	11.83 (0.02)	11.79 (0.06)	11.83 (0.05)	11.79 (0.07)	1m30s	6s	3s	0s	28s	0s	0s
50	16.85 (0.07)	20.01 (0.06)	18.60 (0.32)	19.31 (0.05)	20.05 (0.06)	20.09 (0.05)	20.04 (0.04)	39m37s	8s	4s	0s	1h22m45s	1s	0s
100	-	23.49 (0.06)	21.75 (0.41)	22.90 (0.02)	-	23.69 (0.06)	23.66 (0.04)	-	14s	6s	0s	-	1s	0s
500	-	31.31 (0.06)	29.93 (0.32)	31.35 (0.06)	-	32.14 (0.06)	32.13 (0.07)	-	1m19s	24s	3s	-	2s	0s
1000	-	34.23 (0.08)	33.79 (0.29)	35.09 (0.06)	-	35.82 (0.06)	35.86 (0.04)	-	2m59s	41s	6s	-	4s	0s
2000	-	35.18 (0.14)	37.76 (0.23)	38.84 (0.05)	-	39.63 (0.08)	39.60 (0.05)	-	13m11s	1m28s	13s	-	7s	0s

Table 3. Basket call options on Black–Scholes for different numbers of stocks d.

	price							duration						
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP
5	3.60 (0.05)	3.57 (0.05)	3.49 (0.06)	3.58 (0.03)	3.61 (0.03)	3.62 (0.06)	3.59 (0.02)	13s	6s	2s	0s	2s	0s	0s
10	2.54 (0.04)	2.52 (0.03)	2.45 (0.06)	2.54 (0.04)	2.53 (0.03)	2.53 (0.03)	2.54 (0.01)	30s	6s	1s	0s	7s	0s	0s
50	0.94 (0.01)	1.12 (0.01)	0.83 (0.03)	1.06 (0.01)	1.13 (0.01)	1.15 (0.01)	1.14 (0.01)	8m51s	8s	1s	0s	7m 3s	1s	0s
100	0.51 (0.01)	0.78 (0.01)	0.55 (0.01)	0.75 (0.01)	0.80 (0.01)	0.81 (0.01)	0.81 (0.01)	38m59s	13s	2s	0s	1h21m59s	1s	0s
500	-	0.33 (0.01)	0.24 (0.00)	0.34 (0.00)	-	0.36 (0.00)	0.36 (0.00)	-	1m 7s	7s	1s	-	1s	0s
1000	-	0.22 (0.00)	0.17 (0.00)	0.24 (0.00)	-	0.25 (0.00)	0.26 (0.00)	-	2m24s	14s	2s	-	2s	0s
2000	-	0.13 (0.00)	0.12 (0.01)	0.17 (0.00)	-	0.18 (0.00)	0.18 (0.00)	-	5m35s	25s	7s	-	3s	0s

Comparing the achieved prices of LSM and FQI, we can confirm the claim of [49], that reinforcement learning techniques usually outperform the backward induction in the Markovian setting. RFQI, achieving similar prices as FQI, therefore naturally outperforms RLSM, which achieves similar prices as LSM. A possible explanation for the outperformance of the reinforcement learning algorithm is the following. The backward induction algorithms have approximately N times the number of trainable parameters used in the reinforcement learning algorithms, since a different network is trained for each discretization date. Moreover, for the backward

Table 4. Geometric put options on Black–Scholes and Heston (with variance) for different numbers of stocks d. Here, $r = 2\%$ is used as interest rate.

		price							duration						
model	d	LSM	DOS	NLSM	RLSM	FQI	RFQI	B	LSM	DOS	NLSM	RLSM	FQI	RFQI	B
BlackScholes	5	3.34 (0.04)	3.31 (0.03)	3.29 (0.06)	3.33 (0.04)	3.31 (0.05)	3.35 (0.04)	3.35 (nan)	11s	6s	1s	0s	2s	0s	3m12s
	10	2.37 (0.04)	2.42 (0.02)	2.33 (0.02)	2.40 (0.04)	2.39 (0.03)	2.40 (0.03)	2.40 (nan)	28s	6s	1s	0s	7s	0s	3m12s
	20	1.65 (0.02)	1.71 (0.04)	1.57 (0.04)	1.65 (0.03)	1.73 (0.04)	1.72 (0.02)	1.71 (nan)	1m31s	6s	1s	0s	32s	1s	3m12s
	50	0.91 (0.01)	1.07 (0.02)	0.80 (0.02)	1.03 (0.01)	1.09 (0.02)	1.09 (0.02)	1.09 (nan)	8m26s	10s	2s	0s	7m24s	1s	3m31s
	100	0.50 (0.01)	0.76 (0.01)	0.54 (0.01)	0.73 (0.01)	0.77 (0.01)	0.77 (0.01)	0.78 (nan)	38m37s	16s	2s	0s	1h23m50s	1s	3m31s
Heston	5	2.45 (0.03)	2.44 (0.03)	2.30 (0.06)	2.44 (0.02)	2.44 (0.04)	2.43 (0.03)	-	11s	6s	1s	0s	2s	0s	-
	10	2.00 (0.02)	2.00 (0.02)	1.75 (0.04)	2.00 (0.03)	2.00 (0.02)	2.01 (0.02)	-	29s	6s	2s	0s	7s	0s	-
	20	1.68 (0.02)	1.69 (0.02)	1.21 (0.05)	1.62 (0.05)	1.72 (0.02)	1.71 (0.01)	-	1m31s	7s	2s	0s	32s	1s	-
	50	1.33 (0.02)	1.47 (0.01)	0.83 (0.03)	1.24 (0.01)	1.49 (0.01)	1.48 (0.01)	-	8m31s	7s	3s	0s	7m13s	1s	-
	100	0.88 (0.01)	1.39 (0.01)	0.71 (0.02)	1.18 (0.01)	1.41 (0.01)	1.40 (0.01)	-	41m34s	15s	4s	0s	1h24m11s	1s	-

Table 5. Min put option on Black–Scholes for different numbers of stocks d and varying initial stock price x_0. Here, $r = 2\%$ is used as interest rate.

		price						duration					
d	x_0	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI	RFQI
5	80	35.49 (0.07)	35.48 (0.06)	35.21 (0.12)	35.46 (0.07)	35.53 (0.08)	35.54 (0.05)	11s	10s	3s	0s	3s	0s
	100	19.98 (0.09)	19.96 (0.09)	19.68 (0.07)	19.96 (0.14)	19.97 (0.10)	19.95 (0.09)	11s	9s	3s	0s	3s	0s
	120	7.46 (0.10)	7.36 (0.08)	7.25 (0.07)	7.39 (0.10)	7.45 (0.11)	7.38 (0.10)	11s	6s	1s	0s	2s	0s
10	80	40.22 (0.05)	40.17 (0.05)	39.91 (0.10)	40.21 (0.07)	40.31 (0.07)	40.30 (0.04)	28s	6s	2s	0s	9s	0s
	100	25.74 (0.09)	25.74 (0.10)	25.36 (0.12)	25.76 (0.09)	25.79 (0.10)	25.83 (0.13)	28s	6s	3s	0s	6s	0s
	120	11.98 (0.07)	11.92 (0.09)	11.62 (0.14)	11.94 (0.13)	11.96 (0.10)	12.03 (0.07)	28s	5s	1s	0s	6s	0s
50	80	48.08 (0.05)	48.27 (0.04)	47.03 (0.19)	47.72 (0.03)	48.36 (0.05)	48.34 (0.04)	8m25s	8s	3s	0s	5m20s	1s
	100	35.57 (0.07)	35.80 (0.08)	34.27 (0.40)	35.11 (0.04)	35.91 (0.07)	35.87 (0.08)	8m35s	8s	3s	0s	6m57s	1s
	120	22.93 (0.08)	23.33 (0.07)	21.40 (0.41)	22.50 (0.05)	23.41 (0.06)	23.42 (0.10)	8m28s	8s	2s	0s	6m52s	1s
100	80	49.71 (0.06)	50.93 (0.04)	48.78 (0.26)	50.48 (0.04)	50.93 (0.04)	50.99 (0.03)	39m57s	13s	3s	0s	1h22m58s	1s
	100	37.63 (0.07)	39.11 (0.05)	36.42 (0.80)	38.55 (0.05)	39.11 (0.06)	39.22 (0.04)	40m12s	12s	3s	0s	1h23m26s	1s
	120	25.52 (0.08)	27.31 (0.05)	24.13 (0.52)	26.64 (0.03)	27.28 (0.07)	27.42 (0.05)	40m40s	12s	3s	0s	1h22m53s	1s
500	80	-	55.71 (0.03)	51.51 (0.46)	55.66 (0.03)	-	56.04 (0.03)	-	54s	13s	1s	-	1s
	100	-	45.14 (0.05)	40.06 (1.04)	45.05 (0.02)	-	45.51 (0.03)	-	53s	13s	1s	-	1s
	120	-	34.53 (0.05)	28.39 (0.75)	34.45 (0.05)	-	34.99 (0.02)	-	54s	12s	1s	-	1s
1000	80	-	57.40 (0.03)	53.50 (0.64)	57.52 (0.03)	-	57.84 (0.02)	-	1m36s	21s	3s	-	2s
	100	-	47.24 (0.05)	42.35 (0.63)	47.40 (0.05)	-	47.76 (0.03)	-	1m37s	21s	3s	-	2s
	120	-	37.04 (0.03)	31.10 (0.87)	37.25 (0.04)	-	37.68 (0.03)	-	1m34s	20s	3s	-	2s
2000	80	-	58.59 (0.04)	55.21 (0.67)	59.21 (0.02)	-	59.50 (0.02)	-	3m 1s	30s	6s	-	3s
	100	-	48.72 (0.04)	44.37 (0.61)	49.49 (0.04)	-	49.83 (0.03)	-	2m56s	31s	6s	-	3s
	120	-	38.84 (0.06)	33.31 (0.73)	39.79 (0.04)	-	40.18 (0.04)	-	3m 0s	31s	6s	-	3s

induction algorithms, a different continuation value function is approximated for each date; hence, only the data of this date is used to learn the parameters. In contrast, the reinforcement learning methods train their parameters using the data of all dates. Hence, the reinforcement learning methods use N times the number of data to train $1/N$ times the number of parameters, which seems to lead to better approximations.

We first give a detailed discussion of results for the easy optimal stopping problems, where it is optimal to exercise the option at maturity. Although these optimal stopping problems are less complex, they are still interesting, because a minimal

[3]Due to memory overflow issues, FQI could only be run with 5 instead of 10 parallel runs for larger N; so the computation times are smaller then they would otherwise be, due to more CPU power per run.

TABLE 6. Max call option on Black–Scholes for different numbers of stocks d. Here, $r = 5\%$ is used as interest rate, and $\delta = 10\%$ is used as dividend rate.

	price						duration					
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI	RFQI
5	18.83 (0.17)	18.66 (0.11)	18.62 (0.18)	18.83 (0.12)	18.43 (0.10)	18.83 (0.16)	22s	7s	3s	0s	2s	0s
10	26.67 (0.14)	26.72 (0.17)	26.35 (0.14)	26.60 (0.12)	26.56 (0.13)	26.77 (0.09)	46s	7s	3s	0s	8s	0s
50	43.86 (0.10)	44.52 (0.13)	43.27 (0.33)	43.37 (0.11)	44.66 (0.14)	44.78 (0.12)	10m 5s	10s	4s	0s	7m23s	1s
100	46.62 (0.19)	51.71 (0.09)	49.31 (0.56)	50.61 (0.10)	51.79 (0.17)	52.16 (0.09)	49m27s	15s	5s	0s	1h21m26s	1s
500	-	67.12 (0.09)	62.82 (0.72)	67.02 (0.08)	-	68.48 (0.13)	-	59s	14s	2s	-	2s
1000	-	73.37 (0.12)	69.25 (0.83)	73.85 (0.10)	-	75.31 (0.08)	-	1m52s	26s	4s	-	2s
2000	-	78.17 (0.11)	76.57 (0.83)	80.54 (0.07)	-	81.96 (0.16)	-	5m26s	47s	8s	-	3s

TABLE 7. Min put option on Heston (with variance) for different numbers of stocks d. Here, $r = 2\%$ is used as interest rate.

	price						duration					
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI	RFQI
5	12.34 (0.05)	12.31 (0.05)	12.16 (0.11)	12.29 (0.06)	12.35 (0.09)	12.37 (0.09)	30s	6s	3s	0s	8s	0s
10	16.48 (0.07)	16.52 (0.08)	16.09 (0.13)	16.55 (0.06)	16.64 (0.07)	16.61 (0.08)	1m31s	6s	3s	0s	28s	0s
50	22.86 (0.05)	25.56 (0.04)	24.03 (0.42)	24.85 (0.08)	25.72 (0.03)	25.71 (0.07)	39m57s	9s	4s	0s	1h21m59s	1s
100	-	29.13 (0.04)	27.30 (0.46)	28.50 (0.06)	-	29.33 (0.07)	-	16s	6s	0s	-	1s
500	-	36.26 (0.05)	34.74 (0.31)	36.28 (0.04)	-	36.95 (0.05)	-	1m21s	24s	3s	-	2s
1000	-	38.62 (0.08)	38.19 (0.20)	39.32 (0.03)	-	39.93 (0.05)	-	3m18s	45s	6s	-	4s
2000	-	39.22 (0.13)	41.05 (0.21)	42.22 (0.04)	-	42.81 (0.04)	-	12m51s	1m37s	13s	-	8s

TABLE 8. Max call option on Heston (with variance) for different numbers of stocks d. Here, $r = 5\%$ is used as interest rate, and $\delta = 10\%$ is used as dividend rate.

	price						duration					
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI	RFQI
5	4.88 (0.03)	4.89 (0.03)	4.69 (0.05)	4.83 (0.04)	4.37 (0.06)	4.59 (0.08)	31s	5s	3s	0s	8s	0s
10	7.19 (0.06)	7.20 (0.04)	6.90 (0.07)	7.17 (0.04)	6.63 (0.07)	6.84 (0.06)	1m33s	5s	2s	0s	27s	0s
50	11.68 (0.05)	13.99 (0.07)	12.93 (0.28)	13.70 (0.05)	13.72 (0.09)	13.71 (0.04)	41m 6s	8s	3s	0s	1h22m14s	1s
100	-	17.04 (0.07)	15.94 (0.29)	16.80 (0.03)	-	16.97 (0.05)	-	11s	5s	0s	-	1s
500	-	24.05 (0.05)	22.95 (0.40)	24.35 (0.05)	-	24.70 (0.05)	-	1m19s	23s	3s	-	2s
1000	-	26.86 (0.05)	26.47 (0.39)	27.71 (0.04)	-	28.08 (0.05)	-	2m48s	41s	6s	-	4s
2000	-	28.01 (0.11)	30.12 (0.18)	31.13 (0.05)	-	31.55 (0.07)	-	12m56s	1m30s	14s	-	7s

requirement for the algorithms should be that they perform well in these basic examples. Moreover, a comparison to the reference price is possible. In Table 1 we show results of a max call option on Black–Scholes. For high dimensions ($d \geq 500$), RLSM is about 8 times faster than the fastest baseline NLSM and about 30 times faster than DOS. Moreover, RFQI is about twice as fast as RLSM. For $d = 100$ we also see the large difference in computation time between LSM (respectively, FQI), where the number of basis functions grows quadratically in d, and RLSM (respectively, RFQI), where the number of basis functions does not grow in d. The computed prices of RLSM are at most 2% smaller than those of LSM, and the prices of RFQI are at most 1.2% smaller than those of FQI. For $d \leq 100$ the maximal relative errors compared to the reference prices are 19.7% for LSM, 3.5% for DOS, 9.5% for NLSM, 6.4% for RLSM, 1.2% for FQI and 2.2% for RFQI. For $d \geq 500$

Table 9. Max call option on Black–Scholes for different numbers of stocks d and higher numbers of exercise dates N.

		price							duration						
d	N	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP
10	10	34.33 (0.15)	34.03 (0.17)	33.69 (0.20)	34.28 (0.11)	34.25 (0.19)	34.17 (0.11)	34.26 (0.09)	29s	7s	2s	0s	7s	0s	0s
	50	34.13 (0.12)	34.14 (0.20)	33.96 (0.11)	33.98 (0.08)	34.25 (0.21)	34.15 (0.13)	34.23 (0.11)	2m44s	32s	20s	0s	46s	6s	0s
	100	34.15 (0.14)	34.14 (0.26)	33.98 (0.24)	34.05 (0.15)	34.29 (0.16)	33.97 (0.11)	34.28 (0.10)	5m18s	1m 6s	32s	1s	1m27s	7s	0s
50	10	53.49 (0.10)	53.93 (0.12)	52.15 (0.60)	52.44 (0.21)	54.24 (0.09)	54.23 (0.08)	54.37 (0.09)	8m42s	8s	3s	0s	6m57s	1s	0s
	50	52.82 (0.13)	53.94 (0.18)	53.24 (0.26)	50.85 (0.18)	54.31 (0.14)	53.74 (0.08)	54.46 (0.11)	48m36s	41s	18s	1s	21m15s	7s	0s
	100	52.74 (0.11)	54.09 (0.15)	53.61 (0.18)	50.42 (0.17)	54.15 (0.10)	53.77 (0.12)	54.33 (0.14)	1h37m48s	1m36s	37s	2s	41m 8s	16s	0s
100	10	56.83 (0.18)	61.84 (0.26)	58.99 (0.62)	60.58 (0.10)	62.07 (0.15)	62.32 (0.16)	62.43 (0.08)	40m42s	13s	4s	0s	1h23m28s	1s	0s
	50	-	61.88 (0.06)	60.72 (0.24)	58.68 (0.13)	-	61.66 (0.14)	62.48 (0.07)	-	1m15s	22s	1s	-	8s	0s
	100	-	62.07 (0.11)	61.19 (0.15)	58.26 (0.16)	-	61.79 (0.11)	62.46 (0.04)	-	2m23s	44s	3s	-	15s	0s
500	10	-	78.27 (0.16)	74.23 (1.03)	78.80 (0.10)	-	80.21 (0.13)	80.45 (0.08)	-	53s	12s	1s	-	1s	0s
	50	-	79.14 (0.08)	75.63 (1.07)	76.68 (0.05)	-	79.23 (0.07)	80.44 (0.09)	-	4m59s	1m 4s	8s	-	9s	0s
	100	-	79.44 (0.09)	76.46 (0.41)	76.33 (0.05)	-	79.34 (0.08)	80.47 (0.10)	-	10m12s	2m13s	18s	-	19s	0s

Table 10. Max call option on Black–Scholes for different numbers of stocks d and higher numbers of exercise dates N. Here, $r = 5\%$ is used as interest rate, and $\delta = 10\%$ is used as dividend rate.

		price						duration					
d	N	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI[3]	RFQI
10	10	26.67 (0.14)	26.72 (0.17)	26.35 (0.14)	26.60 (0.12)	26.56 (0.13)	26.77 (0.09)	46s	7s	3s	0s	8s	0s
	50	26.65 (0.12)	26.56 (0.20)	26.42 (0.17)	26.61 (0.13)	26.51 (0.07)	26.69 (0.18)	2m21s	44s	53s	2s	44s	4s
	100	26.68 (0.19)	26.44 (0.14)	26.42 (0.19)	26.59 (0.13)	26.55 (0.14)	26.65 (0.16)	4m46s	2m46s	1m45s	1s	1m25s	8s
50	10	43.86 (0.10)	44.52 (0.13)	43.27 (0.33)	43.37 (0.11)	44.66 (0.14)	44.78 (0.12)	10m 5s	10s	4s	0s	7m23s	1s
	50	43.42 (0.09)	44.50 (0.08)	44.13 (0.19)	42.26 (0.10)	44.68 (0.15)	44.72 (0.12)	43m 5s	1m14s	52s	3s	16m46s	9s
	100	43.21 (0.14)	44.45 (0.15)	44.30 (0.13)	41.89 (0.31)	44.64 (0.17)	44.60 (0.14)	1h25m 4s	2m 0s	1m45s	2s	36m10s	17s
100	10	46.62 (0.19)	51.71 (0.09)	49.31 (0.56)	50.61 (0.10)	51.79 (0.17)	52.16 (0.09)	49m27s	15s	5s	0s	1h21m26s	1s
	50	-	51.73 (0.10)	50.89 (0.21)	49.36 (0.09)	-	51.91 (0.12)	-	52s	33s	1s	-	8s
	100	-	51.72 (0.15)	51.27 (0.12)	48.90 (0.08)	-	51.84 (0.11)	-	1m48s	46s	3s	-	19s
500	10	-	67.12 (0.09)	62.82 (0.72)	67.02 (0.08)	-	68.48 (0.13)	-	59s	14s	2s	-	2s
	50	-	67.48 (0.11)	64.75 (0.50)	65.70 (0.07)	-	68.03 (0.12)	-	2m56s	1m 4s	7s	-	10s
	100	-	67.50 (0.15)	65.55 (0.34)	65.22 (0.07)	-	67.91 (0.10)	-	5m48s	1m52s	16s	-	20s

these errors are 5.6% for DOS, 11% for NLSM, 3% for RLSM and 0.4% for RFQI. The results of Table 2 (max call on Heston with variance) and Table 3 (basket call on Black–Scholes) are similar, except that relative errors become larger in Table 3 for growing d, since the prices become very small. In Figure 1 we plot the price and computation time for at-the-money max call options without dividend on the Heston model when increasing the number of stocks. It is well visible that the computation time of RLSM and RFQI hardly increases, while the prices are similar to the other algorithms.

In the remaining examples, it is in general not optimal to exercise the options at maturity, making the stopping decisions harder and therefore more challenging for the algorithms.

For the geometric put options (Table 4), we do not present dimensions larger than 100, because prices cannot be computed numerically any more. In the Black–Scholes case, the maximal relative errors compared to the reference price are 35.7% for LSM, 2.1% for DOS, 29.9% for NLSM, 6% for RLSM, 1.1% for FQI and 0.5% for RFQI. Again, the prices computed with RLSM (respectively RFQI) are never much smaller than those of LSM (FQI); 0.3% (0.1%) for Black–Scholes and 6% (0.6%) for Heston (with variance). On the Heston model, RFQI, FQI and DOS achieve the highest prices that never deviate more than 1.5% from each other.

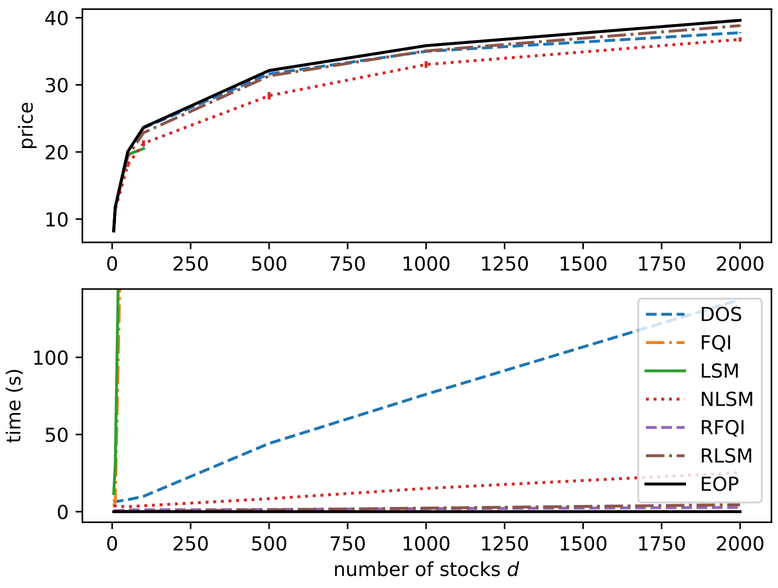

Figure 1. At-the-money max call option without dividend on Heston.

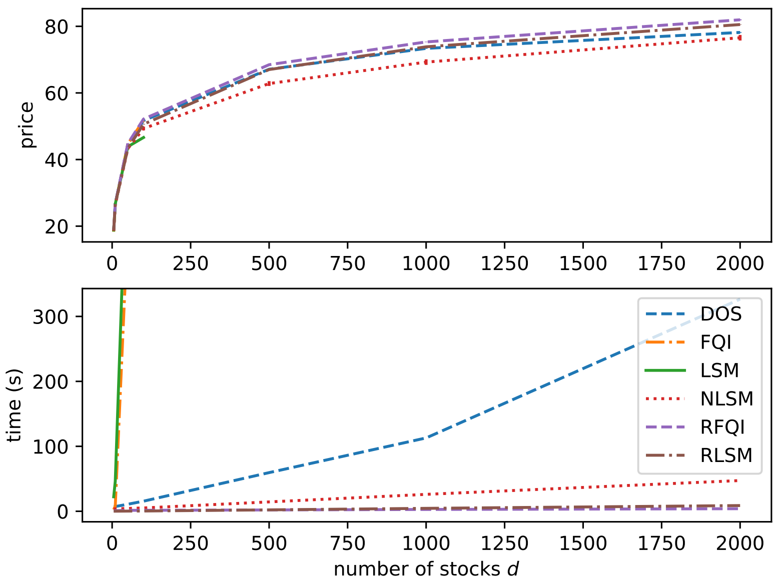

Figure 2. At-the-money max call option with dividend on Black–Scholes.

For the min put option on Black–Scholes (Table 5), RLSM is about 7 times faster than NLSM and more than 30 times faster than DOS for high dimensions. Furthermore, RFQI is again about twice as fast as RLSM. For $d \leq 50$ all algorithms yield very similar prices and for larger d the highest prices are always achieved by RFQI, whereby the prices computed with RFQI never deviate more than 1% from those computed with FQI. Moreover, the prices computed with RLSM are never more than 1.9% smaller than those computed with LSM. In addition, RLSM achieves the second highest prices for high dimensions. For the max call option with dividends on Black–Scholes (Table 6 and Figure 2), the situation is similar. However, the highest prices are always achieved by RFQI and the prices computed with RLSM are at most 1.1% smaller than those of LSM. For the min put option on Heston (with variance) (Table 7) we have similar results as on Black–Scholes, but the prices computed with RLSM (RFQI) are at most 0.5% (0.2%) smaller than those computed with LSM (FQI).

For the max call option with dividend on Heston (with variance) (Table 8), RLSM is about 7 times faster than NLSM and more than 26 times faster then DOS for high dimensions. RFQI is again about twice as fast as RLSM. For $d \in \{5, 10\}$ DOS yields the highest prices, RLSM deviates at most 1.2% from them and RFQI at most 6%. FQI yields lower prices than RFQI. For $d \in \{50, 100\}$, DOS, RLSM and RFQI yield very similar prices, deviating at most 2% from each other. For higher dimensions of $d \geq 500$, RFQI yields the highest prices, and RLSM yields the second highest prices.

When increasing the number of exercise dates for the max call option on Black–Scholes from $N = 10$ to $N \in \{50, 100\}$ (Table 9), the Bermudan option price should become closer to the American option price. The highest prices are achieved either by RFQI, FQI or DOS, with a maximum deviation of less than 1.4% between their results and a maximum deviation from the reference prices of 2.7% for DOS and 1.5% for RFQI. RFQI is more than 30 times faster than DOS for high dimensions. Increasing the number of dates further, the computation time can become a limiting factor for DOS, while this is not the case for RFQI. We see similar results for the more complex max call option on Black–Scholes with dividends (Table 10), where RFQI always achieves the highest price.

7.2.5. *Empirical convergence study.* We confirm the theoretical results of Theorem 2.1 (Figure 3 left) and Theorem 3.1 (Figure 3 right) by an empirical convergence study for a growing number of paths m. For RLSM we also increase the number of hidden nodes K, while they are fixed for RFQI since $d = 5$ is used. For each combination of the number of paths m and the hidden size K, the algorithms are run 20 times and their mean prices with standard deviations are shown. For small m, we see that smaller hidden sizes achieve better prices. This is due to overfitting to the training paths when using larger networks. Regularization techniques like L^1- or L^2-penalization could be used to reduce overfitting for larger networks. However, our results suggest that restricting the hidden size is actually the simplest and best regularization technique, since it additionally leads to lower training times.

7.3. **The non-Markovian case – optimally stopping fractional Brownian motions.**

In order to compare our algorithms on a problem where the underlying process is non-Markovian, we take the example of the fractional Brownian motion $(W_t^H)_{t\geq 0}$ as in [10]. Unlike classical Brownian motion, the increments of fractional Brownian motion need not be independent. Fractional Brownian motion is a continuous centered Gaussian process with covariation function $E\left(W_t^H W_s^H\right) = \frac{1}{2}\left(|t|^{2H} + |s|^{2H} - |t-s|^{2H}\right)$ where $H \in (0, 1]$ is called the Hurst parameter. When the Hurst parameter $H = 0.5$, W^H is a standard Brownian motion; when $H \neq 0.5$, the increments of $(W_t^H)_{t\geq 0}$ are correlated (positively if

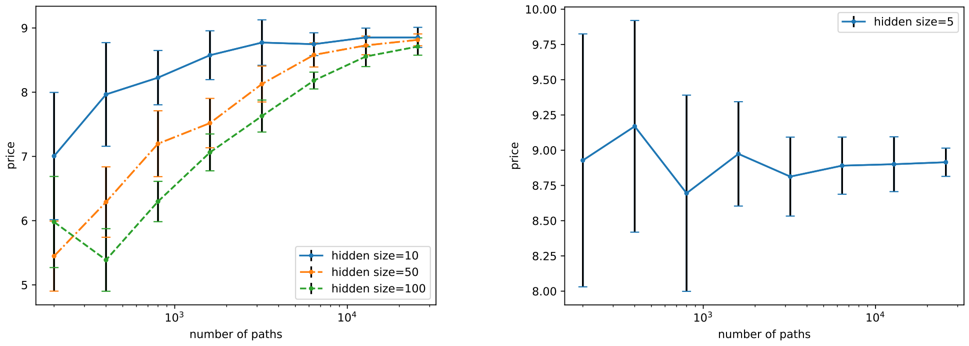

FIGURE 3. Mean ± standard deviation (bars) of the price for a max call on 5 stocks following the Black–Scholes model for RLSM (left) and RFQI (right) for varying the number of paths m and varying for RLSM the number of neurons in the hidden layer K.

$H > 0.5$, and negatively if $H < 0.5$) which means that for $H \neq 0.5$, $(W_t^H)_{t\geq 0}$ is not Markovian [9, 50, 32, 25, 1].

7.3.1. *Stock model, payoffs and baselines.* In this section we use a d-dimensional fractional Brownian motion, with independent coordinates all starting at $X_0 = 0$, as the underlying process $X_t = W_t^H$. In contrast to the price processes we used before, this process can become negative. In the one-dimensional case, we use the identity as "payoff" function $g = \text{id}$ as in [10], which can lead to negative "payoff" values. Moreover, we use the maximum $g(x) = \max(x_1, x_2, \dots, x_d)$ for any $x = (x_1, x_2, \dots, x_d) \in \mathbb{R}^d$ and the mean $g(x) = 1/d \sum_{i=1}^d x_i$ as "payoffs" for higher dimensions, which can also yield negative values. In particular, this setting leads to an optimal stopping problem outside of the standard discretized American option pricing setting. We compare RLSM and RRLSM to DOS and the path version of DOS (denoted pathDOS for our implementation of it and pathDOS-paper for results reported from [10]), where the entire path until the current date is used as input [10]. Moreover, we test RFQI and its recurrent and path versions in this setting.

For two values of the Hurst parameter, the optimal value can be computed explicitly. In particular, for $H = 0.5$ we have a Brownian motion, and therefore the optimal value is 0 . For $H = 1$ we have a fully correlated process (i.e., all information is known after the first step), where the optimal value is approximately 0.39495 [10].

7.3.2. *Results and discussion.* For $d = 1$, we clearly see the outperformance of the algorithms processing information of the path compared to the ones using only the current value as input (Figure 4 top left). Moreover, this application highlights the limitation of reinforcement learning techniques when applied in non-Markovian settings as discussed in [44]. In particular, RFQI, the randomized RNN version of it (RRFQI) and its path- version do not work well in this example (Figure 4 top right). This poor performance was consistent under varying hyper-parameters.

RRLSM achieves very similar results to those reported for pathDOS in [10] with an MSE of 0.0005 between their reported values and ours, while using only 20K instead of 4M paths (Figure 4 bottom). RRLSM needs only $1s$ to be trained, in contrast to $430s$ reported in [10]. The longer training times can partly be explained

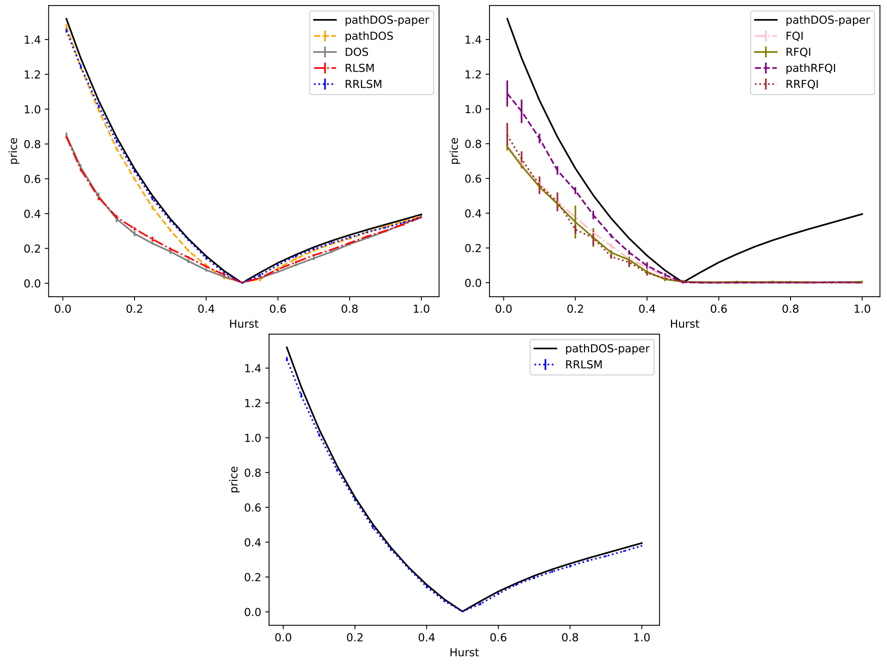

FIGURE 4. Top left: algorithms processing path information outperform. Top right: reinforcement learning algorithms do not work well in non-Markovian cases. Bottom: RRLSM achieves similar results as reported in [10], while using only 20K paths instead of 4M for training, which took only 1s instead of the reported 430s.

by the larger amount of paths used. However, our implementation of pathDOS using the same number of 20 hidden nodes as RRLSM and also being trained on 20K paths (hence completely comparable to the training of RRLSM) takes approximately 175s and achieves slightly worse results than RRLSM (Figure 4 top left) with an MSE of 0.0018. The exact prices displayed in Figure 4 are provided in Appendix C.1.

For higher dimensions, we use the small Hurst parameter $H = 0.05$ for which a big difference between the standard and the path dependent algorithms was visible in the one-dimensional case. RLSM yields very similar prices as DOS, and RRLSM yields very similar prices as pathDOS. However, RLSM and RRLSM are considerably faster than DOS and pathDOS (Table 11).

7.4. **Non-Markovian stock models.** In Section 7.3 we saw that the RL based algorithms do not perform well on problems which are highly path dependent. In this section, we consider "intermediate" problems of typical non-Markovian stock models, where a path dependence exists but where this path dependence is not very strong.

7.4.1. *Heston without variance as input.* First, we revisit the Heston model (11), but this time without feeding the algorithms the variance, which makes it a non-Markovian problem. For the max call (Table 17), min put (Table 18) and max call with dividend (Table 19) options on Heston without variance, all the algorithms yield very similar prices as on Heston with variance (Tables 2, 7 and 8), so we only

Table 11. Identity, maximum and mean on the fractional Brownian motion with $H = 0.05$ and different numbers of stocks d.

		price				duration			
payoff	d	DOS	pathDOS	RLSM	RRLSM	DOS	pathDOS	RLSM	RRLSM
Identity	1	0.67 (0.02)	1.24 (0.01)	0.65 (0.01)	1.24 (0.01)	1 m 15 s	3 m 1 s	0s	1s
Max	5	1.96 (0.01)	2.15 (0.01)	2.00 (0.01)	2.16 (0.01)	3 m 8 s	21 m 46 s	4s	1s
	10	2.34 (0.01)	2.43 (0.01)	2.40 (0.01)	2.43 (0.02)	3 m 49 s	37 m 46 s	4s	2s
Mean	5	0.29 (0.01)	0.53 (0.00)	0.28 (0.01)	0.52 (0.01)	3 m 40 s	21 m 8 s	3s	1s
	10	0.20 (0.01)	0.36 (0.00)	0.21 (0.01)	0.33 (0.01)	3 m 39 s	36 m 1 s	5s	1s

show the tables in Appendix C.2. In particular, this suggests that even though the Heston model is not Markovian without providing the current variance, this does not make a difference for option pricing.

7.4.2. *Rough Heston.* Moreover, we test on the rough Heston model, where the variance itself is path-dependent. This model recently became a very popular choice for modeling financial markets [27, 26, 33]. The rough Heston model [27] is defined as

$$dX_t = (r-\delta)X_t dt + \sqrt{v_t}X_t dW_t,$$

$$v_t = v_0 + \int_0^t \frac{(t-s)^{H-1/2}}{\Gamma(H+1/2)}\kappa(v_\infty - v_s)ds + \int_0^t \frac{(t-s)^{H-1/2}}{\Gamma(H+1/2)}\sigma\sqrt{v_s}dB_s,$$

where $X_0 = x_0$, and the Hurst parameter $H \in (0, 1/2)$ and $(W_t)_{t\geq 0}$ and $(B_t)_{t\geq 0}$ are two d-dimensional Brownian motions correlated with coefficient $\rho \in (-1, 1)$. We choose the drift $r = 5\%$, the dividend rate $\delta = 10\%$, the volatility of volatility $\sigma = 20\%$, the long term variance $v_\infty = 0.01$, the mean reversion speed $\kappa = 2$, the correlation $\rho = -30\%$, the initial stock price $x_0 = 100$ and the initial variance $v_0 = 0.01$ and consider a max call option on the stock price X.

As for the Heston model, also for the rough Heston model there is no significant difference between the computed prices with and without providing the current variance, so we only show prices where the current variance was also fed to the algorithms, which is still a non-Markovian setting. For the max call option on the rough Heston model (with variance) (Table 12), we see that the reinforcement learning based algorithms FQI and RFQI do not work well for $d \in \{5, 10\}$ but perform better for $d \in \{50, 100\}$. Overall, DOS, pathDOS, RLSM and RRLSM achieve very similar prices, never deviating more than 2.2% from each other. In particular, we do not see a better performance of the path dependent algorithms pathDOS and RRLSM compared to DOS and RLSM.

7.5. **Computation of upper bounds.** While this work's focus lies on the lower bound approximations, we conduct a small experiment to show that also the upper bound computation works efficiently with our method. In Table 13 we show mean and standard deviation (over 10 independent runs) of the upper bound approximations for the price of an American option computed with RLSM. Additionally, we show the lower bound and the midpoint (computed as the average of the lower and upper bound). As expected, the upper bound approximations are a bit larger than the lower bound approximations. The same method can also be used to computed

Table 12. Max call option on Rough–Heston for different numbers of stocks d. The interest rate is $r = 5\%$, and the dividend rate is $\delta = 10\%$.

	price								duration							
d	LSM	DOS	pathDOS	NLSM	RLSM	RRLSM	FQI	RFQI	LSM	DOS	pathDOS	NLSM	RLSM	RRLSM	FQI	RFQI
5	6.58 (0.05)	6.56 (0.05)	6.46 (0.06)	6.39 (0.06)	6.50 (0.04)	6.46 (0.04)	6.10 (0.08)	6.33 (0.16)	0s	7s	11s	3s	0s	0s	15s	0s
10	9.41 (0.04)	9.46 (0.04)	9.28 (0.05)	9.27 (0.11)	9.48 (0.04)	9.37 (0.05)	9.19 (0.09)	9.02 (1.18)	1s	7s	13s	3s	0s	0s	37s	0s
50	13.90 (0.07)	16.69 (0.06)	16.47 (0.06)	15.68 (0.32)	16.35 (0.04)	16.37 (0.03)	16.72 (0.07)	16.75 (0.04)	18m51s	9s	39s	4s	0s	0s	1h24m22s	1s
100	-	19.79 (0.05)	19.51 (0.05)	18.39 (0.35)	19.50 (0.04)	19.49 (0.04)	-	19.99 (0.05)	-	13s	1m16s	6s	0s	0s	-	1s

Table 13. Lower, midpoint and upper approximations with RLSM of the price of a max call option on Black–Scholes for different numbers of stocks d and varying initial stock price x_0. The parameters for the stock model are $r = 5\%$, $\delta = 10\%$, $N = 9$, $T = 3$ and $K = 100$. We use $m = 100,000$ paths and 100 neurons for the hidden layer.

d	x_0	price lower	price midpoint	price upper
2	90	7.9772 (0.0512)	8.0077 (0.0362)	8.0382 (0.0619)
2	100	13.7902 (0.0664)	13.8627 (0.0552)	13.9353 (0.0566)
2	110	21.2070 (0.0757)	21.2735 (0.0420)	21.3399 (0.1001)
3	90	11.1869 (0.0373)	11.1937 (0.0466)	11.2005 (0.0603)
3	100	18.5698 (0.0803)	18.6061 (0.0401)	18.6425 (0.0515)
3	110	27.3981 (0.1359)	27.4284 (0.0757)	27.4588 (0.0706)
5	90	16.4518 (0.0645)	16.4995 (0.0410)	16.5473 (0.0479)
5	100	25.9604 (0.0856)	25.9868 (0.0569)	26.0133 (0.0893)
5	110	36.5271 (0.1066)	36.6024 (0.0610)	36.6777 (0.1023)
10	90	25.9808 (0.0923)	26.0235 (0.0568)	26.0662 (0.0805)
10	100	38.0031 (0.0759)	38.0750 (0.0592)	38.1469 (0.0952)
10	110	50.5117 (0.0689)	50.5668 (0.0567)	50.6219 (0.0962)
20	90	37.4659 (0.0944)	37.5135 (0.0737)	37.5611 (0.1440)
20	100	51.3532 (0.1073)	51.3900 (0.0929)	51.4269 (0.1043)
20	110	65.2114 (0.0820)	65.2774 (0.0451)	65.3434 (0.0680)

upper bound approximations for RFQI. However, their quality is relatively sensitive to the number of training iterations and other hyper-parameters; hence, they are not shown here.

7.6. Computation of Greeks.

The Greeks are the sensitivities of the option price to a small change in a given underlying parameter. More precisely, they are partial derivatives of the option prices with respect to different parameters, such as the spot price, time, rate and volatility. We provide experiments (and the code), where we compute the most popular Greeks: delta ($\frac{\partial p_0}{\partial x_0}$), gamma ($\frac{\partial^2 p_0}{\partial x_0^2}$), theta ($\frac{\partial p_0}{\partial t}$), rho ($\frac{\partial p_0}{\partial r}$) and vega ($\frac{\partial p_0}{\partial \sigma}$). The straightforward method to compute them is via the finite difference (FD) method. For theta, rho and vega, the standard forward finite difference method can be used with our algorithms, but they turn out to be unstable for NLSM and DOS. Therefore, we use the central finite difference method, where the exercise boundary is frozen to be the one of the central point, and report

Table 14. Prices and Greeks computed for different strikes K of a 1-dimensional put option on Black–Scholes. For the binomial (B) algorithm, the spacing of the FD method is set to $\varepsilon = 10^{-9}$, which is also used for the other algorithms for delta, theta, rho and vega. For the regression method, $\epsilon = 5$ and a polynomial basis up to degree 9 are used.

		price		delta		gamma		theta	rho	vega
K	algo	FD	regr.	FD	regr.	PDE	regr.			
36	B	0.9192 (–)		-0.1982 (–)		0.0389 (–)		-0.7152 (–)	-6.7085 (–)	10.9100 (–)
36	LSM	0.9024 (0.0086)	0.9006 (0.0094)	-0.1919 (0.0019)	-0.1888 (0.0027)	0.0368 (0.0004)	0.0381 (0.0009)	-0.6615 (0.0068)	-6.8267 (0.0497)	10.7107 (0.0918)
36	RLSM	0.9020 (0.0069)	0.9073 (0.0110)	-0.1940 (0.0019)	-0.1947 (0.0026)	0.0371 (0.0003)	0.0379 (0.0010)	-0.6665 (0.0063)	-6.8241 (0.0639)	10.7590 (0.0629)
36	FQI	0.8504 (0.0067)	0.8642 (0.0116)	-0.1777 (0.0016)	-0.1770 (0.0015)	0.0329 (0.0003)	0.0329 (0.0011)	-0.5753 (0.0047)	-7.7168 (0.0676)	10.3836 (0.0841)
36	RFQI	0.9005 (0.0087)	0.8766 (0.0138)	-0.1924 (0.0029)	-0.1847 (0.0041)	0.0368 (0.0005)	0.0369 (0.0009)	-0.6612 (0.0105)	-6.8282 (0.0891)	10.7093 (0.0880)
36	NLSM	0.8948 (0.0238)	0.8817 (0.0151)	-0.2010 (0.0174)	-0.1905 (0.0038)	0.0368 (0.0013)	0.0380 (0.0006)	-0.6427 (0.0124)	-6.9383 (0.0625)	10.6464 (0.1207)
36	DOS	0.9068 (0.0093)	0.9081 (0.0106)	-0.1957 (0.0024)	-0.1940 (0.0033)	0.0359 (0.0013)	0.0378 (0.0012)	-0.6251 (0.0393)	-7.0119 (0.3001)	10.7249 (0.1419)
40	B	2.3196 (–)		-0.4047 (–)		0.0611 (–)		-0.8446 (–)	-11.2405 (–)	14.7517 (–)
40	LSM	2.2916 (0.0107)	2.2715 (0.0116)	-0.3930 (0.0022)	-0.4049 (0.0053)	0.0579 (0.0003)	0.0637 (0.0009)	-0.7711 (0.0051)	-11.6289 (0.0573)	14.6886 (0.0548)
40	RLSM	2.2897 (0.0095)	2.2970 (0.0135)	-0.3984 (0.0052)	-0.4076 (0.0039)	0.0583 (0.0006)	0.0610 (0.0012)	-0.7721 (0.0067)	-11.6357 (0.1092)	14.7026 (0.0467)
40	FQI	2.2593 (0.0121)	2.2078 (0.0161)	-0.3661 (0.0017)	-0.3934 (0.0038)	0.0541 (0.0003)	0.0655 (0.0013)	-0.7179 (0.0057)	-12.6145 (0.0856)	14.7479 (0.0670)
40	RFQI	2.2239 (0.0407)	2.1252 (0.0374)	-0.3623 (0.0140)	-0.3654 (0.0123)	0.0529 (0.0024)	0.0570 (0.0024)	-0.6897 (0.0415)	-12.9713 (0.6541)	14.6794 (0.0831)
40	NLSM	2.2586 (0.0149)	2.2599 (0.0236)	-0.3830 (0.0136)	-0.4034 (0.0035)	0.0565 (0.0013)	0.0620 (0.0012)	-0.7529 (0.0166)	-11.7895 (0.2469)	14.6545 (0.0970)
40	DOS	2.2884 (0.0102)	2.2963 (0.0099)	-0.4031 (0.0037)	-0.4071 (0.0039)	0.0587 (0.0004)	0.0611 (0.0006)	-0.7727 (0.0055)	-11.5870 (0.1041)	14.7045 (0.0579)
44	B	4.6629 (–)		-0.6654 (–)		0.0779 (–)		-0.6169 (–)	-11.8974 (–)	12.8541 (–)
44	LSM	4.6141 (0.0175)	4.6177 (0.0120)	-0.6476 (0.0037)	-0.6693 (0.0051)	0.0743 (0.0003)	0.0676 (0.0012)	-0.5468 (0.0055)	-12.8537 (0.1337)	13.1799 (0.1129)
44	RLSM	4.6167 (0.0205)	4.6407 (0.0178)	-0.6541 (0.0067)	-0.6699 (0.0036)	0.0746 (0.0005)	0.0641 (0.0022)	-0.5400 (0.0034)	-12.7787 (0.2212)	13.0670 (0.1566)
44	FQI	4.5366 (0.0221)	4.5476 (0.0137)	-0.5962 (0.0039)	-0.6766 (0.0054)	0.0695 (0.0005)	0.0710 (0.0013)	-0.5216 (0.0066)	-15.2857 (0.1248)	14.3877 (0.0471)
44	RFQI	4.3469 (0.0708)	4.1557 (0.0415)	-0.5463 (0.0073)	-0.5791 (0.0059)	0.0607 (0.0023)	0.0687 (0.0019)	-0.3688 (0.0534)	-19.9917 (1.0327)	15.6835 (0.1199)
44	NLSM	4.5820 (0.0211)	4.5996 (0.0531)	-0.6553 (0.0150)	-0.6713 (0.0048)	0.0742 (0.0008)	0.0658 (0.0029)	-0.5268 (0.0182)	-13.1123 (0.9381)	12.9567 (0.5430)
44	DOS	4.6206 (0.0126)	4.6536 (0.0113)	-0.6580 (0.0046)	-0.6695 (0.0042)	0.0747 (0.0003)	0.0641 (0.0016)	-0.5350 (0.0050)	-12.8331 (0.2236)	13.0304 (0.1295)

results only with this method. For computing delta we use the same method, since the others are unstable for all algorithms. Moreover, the computation of gamma, as a second derivative, turns out to be unstable when computed with the second order finite difference method, even when using the same technique as for delta. Therefore, we use two alternative ways to circumvent this instability. The first one (PDE method) is specific to the case of an underlying Black–Scholes model, where the Black–Scholes PDE

$$\frac{\partial p_0}{\partial t} + \frac{1}{2}\sigma^2 x_0^2 \frac{\partial^2 p_0}{\partial x_0^2} + r x_0 \frac{\partial p_0}{\partial x_0} - r p_0 = 0$$

can be used to express gamma in terms of the price, delta and theta. The second one (regression method) is the "naive method" suggested in [48, Section 3.1]. It fits a polynomial regression to option prices achieved when distorting the initial price x_0 by a noise term $\xi \sim N(0, \epsilon^2)$. Then, the price, delta and gamma can easily be computed by evaluating the fitted regression and its first and second derivative (which are easily computed, since polynomial regression is used) at the initial price x_0. The parameter ϵ controls the variance-bias trade-off and has to be chosen by hand. However, the authors also suggested a 2-step method that reduces variance and bias, where this parameter is chosen automatically.

For comparability, we compute the Greeks for the same example as in [48]. In particular, we consider a put option on $d = 1$ stock following a Black–Scholes model with initial price $x_0 = 40$, strike $K \in \{36, 40, 44\}$, rate $r = 6\%$, volatility $\sigma = 20\%$, $N = 10$ equidistant dates, maturity $T = 1$ and $m = 100,000$ paths. The models are run 10 times, and means and standard deviations are reported in Table 14. The price, delta and gamma are computed with both the finite difference (and PDE) and the regression method. As reference we use the binomial model with $N = 50,000$ equidistant dates, for which only the finite difference (and PDE) method is used. The hidden size was set to 10 to account for the smaller input dimension, and the payoff was not used as input except for DOS, where it improved the results considerably. For RLSM the activation function was changed to Softplus, since this worked best, although all other tested activation functions did also yield good results. Overall, RLSM and DOS with the regression method achieve the best results. Furthermore, we highlight, that the time advantage of RLSM and RFQI also comes into play for the computation of Greeks, when increasing the dimension d.

For RLSM (with Softplus activation) we additionally show stability plots of the Greeks with respect to the spot price. In particular we use the same setting as before of a put option on $d = 1$ stock following a Black–Scholes model with rate $r = 6\%$, $N = 10$ dates and $m = 100,000$ paths. However, we fix the strike $K = 40$ and vary the spot price $x_0 \in [20, 60]$. Moreover, we vary the volatility $\sigma \in \{0.1, 0.2, 0.3\}$ and the maturity $T \in \{0.5, 1, 2\}$. For each combination, we run the algorithm 5 times and plot the median of the results in Figure 5. Up to small numerical instabilities, the resulting curves are smooth, as known from theory. We observe the same qualitative behavior of the Greeks as was shown in [22, Section 5.2 - 5.6].

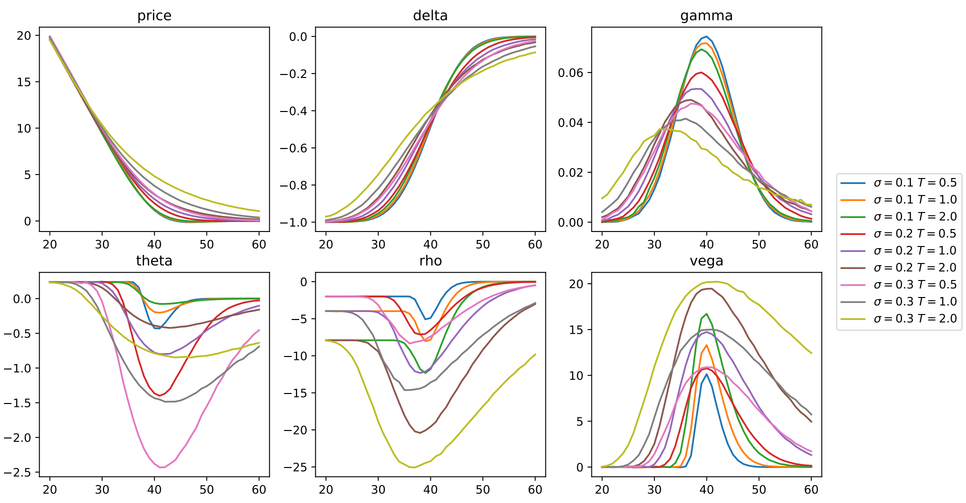

FIGURE 5. Median of the price and Greeks computed with RLSM plotted against the spot price x_0 for different volatilities σ and maturities T. The price, delta and gamma are computed with the regression method with $\epsilon = 5$ and a polynomial basis up to degree 2.

7.7. **Discussion of the sensitivity to the randomness in the hidden layers.** We perform a test specifically designed to study the model's sensitivity to the randomness of the weights in the hidden layers. In our previous tests in this paper, we performed 10 runs, where a different set of paths and different weights of the hidden layer were chosen for each run. In order to test the sensitivity to the randomness of the weights, we perform an experiment with 10 runs, where only the set of hidden weights are different for each run, while the paths are the same.

We compare RLSM and NLSM in the setting of a 1-dimensional Black-Scholes call option with spot $x_0 = 100$ and strike $K = 100$, where we use $100,000$ paths and 10 exercises dates with 20 hidden nodes and either 10, 30 or 50 epochs of training for NLSM.

In order to have a fair comparison, we do not fix the initial weights of NLSM, as it would be equivalent to reusing the same random weights for RLSM in each run, with the possibility of having a good or bad initialization. Hence, similar to RLSM's sensitivity to the randomness of the weights in the hidden layer, NLSM is sensitive to the randomness in the initialization of the weights (of the hidden layer). In order to reduce this sensitivity in the algorithms, one should always take the average of several runs with different sets of weights (and paths). This can be easily done in parallel in order to reduce the computation time. The results of this sensitivity analysis are given in Table 15. We see that the sensitivity of NLSM to the randomness of the initialization depends on the number of epochs of the training, becoming smaller with longer training.

In order to further reduce the sensitivity of RLSM to the randomness of the hidden layer weights, we propose a variant of it, which we call RLMSreinit. Instead of using the same random weights for each date, we use different ones, which has an averaging effect and therefore reduces the variance in multiple runs.

algo	#epochs	price	delta	gamma	theta	rho	vega	duration
NLSM	10	8.8961 (0.0356)	0.5845 (0.0018)	0.0194 (0.0001)	-4.8682 (0.0142)	46.0934 (1.2538)	39.4202 (0.0613)	7.86s
NLSM	30	8.9175 (0.0171)	0.5836 (0.0008)	0.0194 (0.0001)	-4.8723 (0.0206)	47.2922 (1.2718)	39.3781 (0.0640)	19.53s
NLSM	50	8.9178 (0.0160)	0.5835 (0.0006)	0.0194 (0.0001)	-4.8780 (0.0166)	46.6717 (1.6730)	39.3755 (0.1004)	35.03s
RLSM		8.9439 (0.0179)	0.5828 (0.0004)	0.0195 (0.0001)	-4.8952 (0.0153)	48.0459 (0.7537)	39.3425 (0.0951)	1.37s
RLSMreinit		8.9425 (0.0081)	0.5828 (0.0002)	0.0195 (0.0000)	-4.8887 (0.0078)	47.7329 (0.3765)	39.3477 (0.0573)	1.52s

Table 15. Prices and Greeks for NLSM (with different number of training epochs), RLSM and RLSMreinit with standard deviations computed over 10 runs with different initializations on the same set of paths.

8. Conclusion.

Based on a broad study of machine learning based approaches to approximate the solution of optimal stopping problems, we introduced two simple and powerful approaches, RLSM and RFQI. As state-of-the-art algorithms, they are very simple to implement and have convergence guarantees. Moreover, similarly to the neural network methods, they are easily scalable to high dimensions and there is no need to choose basis functions by hand. Furthermore, in our empirical study we saw that RLSM and RFQI are considerably faster than existing algorithms for high dimensional problems: in particular, up to 2400 and 4800 times faster than LSM and FQI, respectively, with basis functions of order 2, 5 to 16 times faster than NLSM and 20 to 66 times faster than DOS.

In our Markovian experiments, RFQI often achieves the best results or, if not, usually is very close to the best performing baseline method under consideration, reconfirming that reinforcement learning methods surpass backward induction methods.

In our non-Markovian experiments on fractional Brownian motion, our randomized recurrent neural network algorithm RRLSM achieves similar results as the path version of DOS, while requiring less training data and being much faster. However, this example also brought up the limitations of reinforcement learning based approaches, in particular of RFQI, which do not work well in those non-Markovian experiments.

In our non-Markovian experiments on rough Heston, we concluded that there is no need of using a recurrent neural network, since RLSM has similar results as RRLSM. This is also the case with DOS and pathDOS.

Overall, the speed of our algorithms is very promising for applications in high dimensions and with many discretization times, where existing methods might become impractical and where our methods show very reliable performance. To summarize, we suggest to use RFQI for Markovian problems, especially in high-dimensional settings; RLSM for low-dimensional settings or when computing Greeks and upper bounds; RLSM for non-Markovian processes which do not have a strong path-dependence, such as the stock price of rough Heston; and finally, RRLSM for non-Markovian processes which have a strong path-dependence like fractional Brownian motion.

Appendix A. Convergence of the randomized least squares Monte Carlo (RLSM).

We first introduce some technical notation that will be helpful for the proofs. Then, we describe the steps from the theoretical idea of RLSM to its implementable version that was presented in Section 2.7. These descriptions and proofs are based on [64, 20], and in particular, our theoretical results are a direct consequence of these works and the universal approximation theorem of [71]. Nevertheless, we give a detailed description here for completeness.

A.1. Definitions.

We assume having a sequence of infinitely many *random* basis functions $\phi = (\phi_k)_{k\geq 1}$, where each ϕ_k is of the form

$$\phi_k : \mathbb{R}^d \to \mathbb{R}, x \mapsto \phi_k(x) := \sigma(\alpha_k^\top x + \beta_k),$$

with σ a bounded activation function, $\alpha_k \in \mathbb{R}^d$ and $\beta_k \in \mathbb{R}$. The parameters α_k and β_k have independent and identically distributed (i.i.d.) entries with a standard Gaussian distribution, hence the name *random* basis functions. With $(\tilde{\Omega}, \tilde{\mathcal{F}}, \tilde{\mathbb{P}})$ we denote the probability space on which the random weights are defined. For each $K \in \mathbb{N}$ we define the operator Φ_K acting on $\theta = (\theta_1, \ldots, \theta_K) \in \mathbb{R}^K$ by

$$(\Phi_K\theta)(x) := \theta^\top \phi(x) := \sum_{k=1}^{K} \theta_k \phi_k(x).$$

In particular, Φ_K is the operator producing a linear combination of the first K random basis functions. We assume having a Markovian, discrete time stochastic process $X = (X_0, \ldots, X_N)$ defined on a filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_n)_{n=0}^N, \mathbb{P})$. In particular, each X_n is an $\mathcal{F}_n$-measurable random variable. We assume that there exists an absolutely continuous measure $\mathbb{Q} \ll \mathbb{P}$, the pricing measure, and that the distribution of X_n under $\mathbb{Q}$ is π_n. For expectations with respect to these random variables under $\mathbb{Q}$, we write $\mathbb{E}[\cdot]$. For $0 \leq n \leq N$ we use the norm

$$\|f\|_{\pi_n}^2 := \mathbb{E}[|f(X_n)|_2^2] = \int_{\mathbb{R}} |f(x)|_2^2 d\pi_n(x),$$

where $|\cdot|_2$ is the Euclidean norm, and f is a measurable function. We introduce the operators E_n and Π_n^K defined by

$$(E_n J)(x) := \mathbb{E}[J(X_{n+1})|X_n = x],$$
$$(\Pi_n^K J) := \arg\min_{\Phi_K\theta} \|J - \Phi_K\theta\|_{\pi_n},$$

for $J \in L^2(\pi_n)$. With $\hat{E}_n$ we denote the one-sample approximation of E_n, i.e., $(\hat{E}_n J)(X_n) = J(X_{n+1})$, which is better understood in terms of a realization of $x = (x_0, \dots, x_N)$ of X as $(\hat{E}_n J)(x_n) = J(x_{n+1})$. Moreover, $\hat{\Pi}_n^K$ is the Monte Carlo approximation of Π_n^K, i.e., if $x_n^1, \dots, x_n^m$ are i.i.d. samples of π_n, then $(\hat{\Pi}_n^K J) := \arg\min_{\Phi_K\theta} \frac{1}{m}\sum_{i=1}^m \left(J(x_n^i) - (\Phi_K\theta)(x_n^i)\right)^2$. In the following, we write Π_n and $\hat{\Pi}_n$ whenever K is fixed.

The payoff at any exercise time n is given by $g(X_n)$, and we assume that they are square integrable, i.e., $\|g(X_n)\|_{\pi_n} < \infty$.

A.2. **Theoretical description of RLSM.** We first introduce the *exact algorithm* to compute the continuation value and then give definitions of the 2-step approximation of this exact algorithm. The first step is to introduce projections on the subspace of functions spanned by Φ_K, while assuming that (conditional) expectations can be computed exactly. We call this the *idealized algorithm*. We remark that also the projection itself is based on minimizing an expectation. The second step is to introduce Monte Carlo and one-sample approximations of the projections and (conditional) expectations using m sample paths. This we call the *implementable algorithm*, since it can actually be implemented. Our goal is then to show that the price computed with those two approximation steps converges to the true price, when K and m increase to infinity.

A.2.1. *Exact algorithm.* The *continuation value* is the expected discounted payoff at the current time conditioned on a decision not to exercise the option now. The *exact algorithmic* definition of the continuation value is defined backward step-wise as in [64] as

$$\begin{cases} Q_{N-1} & := \alpha E_{N-1} g, \\ Q_n & := \alpha E_n \max(g, Q_{n+1}). \end{cases} \tag{12}$$

A.2.2. *Idealized algorithm.* Our *idealized algorithm* to compute the continuation value, written similar as in [64], is defined for fixed K as

$$\begin{cases} \tilde{Q}_{N-1}^K & := \alpha E_{N-1} P_N^K, \\ \tilde{Q}_n^K & := \alpha E_n P_{n+1}^K, \end{cases} \tag{13}$$

where

$$\begin{cases} P_N^K & := g, \\ P_n^K & := g\mathbf{1}_{g \geq \alpha\Pi_n^K E_n P_{n+1}^K} + \alpha E_n P_{n+1}^K \mathbf{1}_{g < \alpha\Pi_n^K E_n P_{n+1}^K}. \end{cases}$$

In particular, P_n^K can be interpreted as the choice of the algorithm, at time step n, to either execute and take the payoff or to continue with the expected discounted future payoff. We drop the superscript K whenever it is clear from the context which K is meant. We see from this equation, that the difference from the idealized algorithm in [64, described in (1) and before Theorem 1] is that we use the $\tilde{Q}_{n+1}$

instead of its linear approximation with the random basis functions $\Pi_n \tilde{Q}_{n+1}$, if we decide to continue. However, the decision to continue or to stop is still based on the approximation $\Pi_n \tilde{Q}_{n+1}$, as it is also the case in the idealized algorithm [64]. If the linear approximation is exact, both algorithms produce the same output, but if it is not exact, our algorithm uses a better approximation of the continuation value.

A.2.3. *Implementable algorithm.* Finally, we define our *implementable algorithm* to compute the continuation value, which is an approximation of the idealized algorithm using the approximations $\hat{E}_n$ and $\hat{\Pi}_n^K$ as

$$\begin{cases} \hat{\tilde{Q}}_{N-1}^K & := \alpha \hat{E}_{N-1} \hat{P}_N^K, \\ \hat{\tilde{Q}}_n^K & := \alpha \hat{E}_n \hat{P}_{n+1}^K, \end{cases} \tag{14}$$

where

$$\begin{cases} \hat{P}_N^K & := g, \\ \hat{P}_n^K & := g \mathbf{1}_{g \geq \alpha \hat{\Pi}_n^K \hat{E}_n \hat{P}_{n+1}^K} + \alpha \hat{E}_n \hat{P}_{n+1}^K \mathbf{1}_{g < \alpha \hat{\Pi}_n^K \hat{E}_n \hat{P}_{n+1}^K}. \end{cases}$$

Also here, we drop the superscript K whenever it is clear from the context which K is meant.

A.3. **Preliminary results.** The following result is similar to [71, Theorem 3] and states, that the error of the approximation of any integrable function by randomized neural networks converges $\tilde{\mathbb{P}}$-a.s. to 0 as the number of hidden nodes goes to infinity, where $\tilde{\mathbb{P}}$ is the probability measure associated with the random weights. While [71, Theorem 3] shows universal approximation in L^p-norm with respect to the Lebesgue integral on a compact subset, we show it with respect to a probability measure on the entire space. We note that our result also holds when replacing the probability measure with a finite measure. In particular, our result implies the result of [71], by using the finite measure that coincides with the Lebesgue measure on the respective compact set and vanishes outside. For completeness, we give an independent proof of our result here.

Theorem A.1. *Let $0 \leq n \leq N-1$ and J be a square integrable function, i.e. $\|J\|_{\pi_n} < \infty$. Then,*

$$\|\Pi_n^K J - J\|_{\pi_n} \xrightarrow[K \to \infty]{\tilde{\mathbb{P}}\text{-}a.s.} 0.$$

Lemma A.2. *Let $\mathcal{X}, \mathcal{Y}$ be normed spaces and μ be a probability measure on $\mathcal{X}$ with its Borel σ-Algebra. Let $J : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ be a bounded function such that for each $C > 0$ and each $x \in \mathcal{X}$ with $\|x\| < C$ the function $y \mapsto J(x,y)$ is Lipschitz continuous with Lipschitz constant L_C (depending only on C but not on x). Then, for any $\epsilon > 0$ and $y \in \mathcal{Y}$, there exists an open neighborhood $O(y,\epsilon) \subset \mathcal{Y}$ such that $y \in O(y,\epsilon)$, and for every $\tilde{y} \in O(y,\epsilon)$ we have*

$$\int_{\mathcal{X}} |J(x,y) - J(x,\tilde{y})|^2 d\mu(x) < \epsilon.$$

Proof. Since J is bounded, there exists M such that $|J| < M$. Since μ is finite, there exists some C such that $\mu(\{x \in \mathcal{X} | \|x\| \geq C\}) < \frac{\epsilon}{8M^2}$. Hence, for any $\tilde{y} \in \mathcal{Y}$

$$\int_{\mathcal{X}} |J(x,y) - J(x,\tilde{y})|^2 \mathbf{1}_{\|x\| \geq C} \, d\mu(x) < \int_{\mathcal{X}} (2M)^2 \mathbf{1}_{\|x\| \geq C} \, d\mu(x) < \epsilon/2.$$

Let us choose $O(y,\epsilon) := B\left(y, \sqrt{\frac{\epsilon}{2L_C^2}}\right)$, the open ball with radius $\sqrt{\frac{\epsilon}{2L_C^2}}$ and center y. Then, for any x with $\|x\| < C$ and $\tilde{y} \in O(y,\epsilon)$, we have $|J(x,y) - J(x,\tilde{y})| < L_C\|y-\tilde{y}\| < \sqrt{\epsilon/2}$. Therefore,

$$\int_{\mathcal{X}} |J(x,y) - J(x,\tilde{y})|^2 \mathbf{1}_{\|x\|<C}\, d\mu(x) < \int_{\mathcal{X}} \epsilon/2\, d\mu(x) < \epsilon/2.$$

Together, this yields the result. □

We first prove the following weaker version of the statement of Theorem A.1.

Lemma A.3. *Let $1 \le n \le M$ and J be an integrable function, i.e., $\|J\|_{\pi_n} < \infty$. Then,*

$$\|\Pi_n^K J - J\|_{\pi_n} \xrightarrow[K\to\infty]{\tilde{\mathbb{P}}} 0.$$

Proof. We fix $\varepsilon > 0$. We have to show that

$$\lim_{K\to\infty} \tilde{\mathbb{P}}\left[\|\Pi_n^K J - J\|_{\pi_n} > \varepsilon\right] = 0.$$

By the universal approximation theorem [41, Theorem 1], there exists a 1-hidden layer neural network $\hat{J}$ with n_1 hidden neurons such that $\|\hat{J} - J\|_{\pi_n} < \varepsilon/2$. Without loss of generality, we assume that the bias of the last layer is 0, which can be established by introducing another hidden neuron which is constant as a function of its input. We notice that also $\Phi_K\theta$ is a 1-layer neural network with K hidden nodes (and the same activation function as $\hat{J}$), where the weights of the input layer are i.i.d. sampled of a normal distribution and then fixed, and the weights of the output layer are θ. Let $\theta^\star \in \mathbb{R}^{n_1}$ be the weights of the output layer of $\hat{J}$. For each of the hidden nodes $1 \le \nu \le n_1$ of $\hat{J}$, we denote the mapping from the input to this hidden node by $\hat{J}^h_\nu$. Let W_ν be the weights defining $\hat{J}^h_\nu$, and denote by Ψ the operator mapping the weights to the corresponding neural network layer, such that $\Psi W_\nu = \hat{J}^h_\nu$. We know from [39] that $\hat{J}^h_\nu$ is Lipschitz continuous w.r.t. the weights for a bounded input $\|x\| \le N$. Moreover, since the activation function is bounded, so is $\hat{J}^h_\nu$. Therefore, by Lemma A.2 there exists an open neighbourhood $\mathcal{W}_\nu$ of W_ν such that for all $W \in \mathcal{W}_\nu$ we have

$$\|\Psi W - \Psi W_\nu\|_{\pi_n} < \tfrac{\varepsilon}{2\sqrt{n_1}|\theta^\star|_2}.$$

For any non-empty open set, the probability that a standard Gaussian random variable lies in this open set is positive. Let V_k be the weights of the k-th random map ϕ_k, i.e., $\phi_k = \Psi V_k$, and note that V_k is a vector of i.i.d. standard Gaussian random variables. Since $\mathcal{W}_\nu$ is open, we therefore have that $\tilde{\mathbb{P}}[V_k \in \mathcal{W}_\nu] > 0$. By independence of the weights, we have that with probability 1 each $\hat{J}^h_\nu$ is approximated well by some ϕ_k when $K \to \infty$. Indeed, let $K = n_1\tilde{K}$, and then we have

$$\begin{aligned}
&\tilde{\mathbb{P}}\left[\forall 1 \le \nu \le n_1 \ \exists 1 \le k \le K : V_k \in \mathcal{W}_\nu\right] \\
&\ge \tilde{\mathbb{P}}\left[\forall 1 \le \nu \le n_1 \ \exists (\nu-1)\tilde{K} < k \le \nu\tilde{K} : V_k \in \mathcal{W}_\nu\right] \\
&= \prod_{\nu=1}^{n_1} \tilde{\mathbb{P}}\left[\exists (\nu-1)\tilde{K} < k \le \nu\tilde{K} : V_k \in \mathcal{W}_\nu\right] \\
&= \prod_{\nu=1}^{n_1} \left(1 - \tilde{\mathbb{P}}\left[\forall (\nu-1)\tilde{K} < k \le \nu\tilde{K} : V_k \notin \mathcal{W}_\nu\right]\right) \\
&= \prod_{\nu=1}^{n_1} \left(1 - \tilde{\mathbb{P}}\left[V_1 \notin \mathcal{W}_\nu\right]^{\tilde{K}}\right) \xrightarrow{\tilde{K}\to\infty} 1,
\end{aligned}$$

where we used in lines 3 and 5 independence of the weights and for the limit that $\tilde{\mathbb{P}}[V_1 \notin \mathcal{W}_\nu] < 1$. We define $\tilde{\theta}^\star \in \mathbb{R}^K$ to have the k-th coordinate equal to $\theta^\star_\nu$ if $k = k(\nu) := \arg\min_j \|\phi_j - \hat{J}^h_\nu\|_{\pi_n}$ or 0 otherwise. Here we assume without loss of generality that all $k(\nu)$ are different (if they are not, the weights are summed up). Then, we have

$$\begin{aligned}
\|\Pi^K_n J - J\|_{\pi_n} \le \|\Phi_K\tilde{\theta}^\star - J\|_{\pi_n} &\le \|\Phi_K\tilde{\theta}^\star - \hat{J}\|_{\pi_n} + \|\hat{J} - J\|_{\pi_n} \\
&\le \|\Phi_K\tilde{\theta}^\star - \hat{J}\|_{\pi_n} + \varepsilon/2,
\end{aligned}$$

and therefore

$$\begin{aligned}
\tilde{\mathbb{P}}\left[\|\Pi^K_n J - J\|_{\pi_n} > \varepsilon\right] &\le \tilde{\mathbb{P}}\left[\|\Phi_K\tilde{\theta}^\star - \hat{J}\|_{\pi_n} > \varepsilon/2\right] \\
&= \tilde{\mathbb{P}}\left[\|(\Psi V_k)_{k=1}^K\tilde{\theta}^\star - (\Psi W_\nu)_{\nu=1}^{n_1}\theta^\star\|_{\pi_n} > \varepsilon/2\right].
\end{aligned}$$

Now, we notice that $\tilde{\theta}^\star_k$ is 0 unless $k = k(\nu)$ for some $1 \le \nu \le n_1$. Hence,

$$\begin{aligned}
\tilde{\mathbb{P}}\left[\|\Pi^K_n J - J\|_{\pi_n} > \varepsilon\right] &\le \tilde{\mathbb{P}}\left[\|(\Psi V_{k(\nu)})_{\nu=1}^{n_1}\theta^\star - (\Psi W_\nu)_{\nu=1}^{n_1}\theta^\star\|_{\pi_n} > \varepsilon/2\right] \\
&\le \tilde{\mathbb{P}}\left[\|(\Psi V_{k(\nu)})_{\nu=1}^{n_1} - (\Psi W_\nu)_{\nu=1}^{n_1}\|_{\pi_n} > \tfrac{\varepsilon}{2|\theta^\star|_2}\right] \\
&\le \tilde{\mathbb{P}}\left[\exists 1 \le \nu \le n_1 : \|\Psi V_{k(\nu)} - \Psi W_\nu\|_{\pi_n} > \tfrac{\varepsilon}{2\sqrt{n_1}\|\theta^\star\|}\right] \\
&= 1 - \tilde{\mathbb{P}}\left[\forall 1 \le \nu \le n_1 \exists 1 \le k \le K : V_k \in \mathcal{W}_\nu\right] \xrightarrow{K\to\infty} 0.
\end{aligned}$$

For the second inequality we used the Cauchy-Schwarz inequality and that $\|\theta^\star\|_{\pi_n} = |\theta^\star|_2$. In the last equality we used that $k(\nu)$ is chosen such that the distance between $\Psi V_{k(\nu)}$ and ΨW_ν is minimized. □

Proof of Theorem A.1. By Lemma A.3 we know that $\|\Pi^K_n J - J\|_{\pi_n} \xrightarrow[K\to\infty]{\tilde{\mathbb{P}}} 0$, which implies that there exists a subsequence $(K_m)_{m\ge1}$ s.t. $\|\Pi^{K_m}_n J - J\|_{\pi_n} \xrightarrow[m\to\infty]{\tilde{\mathbb{P}}\text{-a.s.}} 0$. Let $\hat{\Omega} \subset \tilde{\Omega}$ with $\tilde{\mathbb{P}}(\hat{\Omega}) = 1$ be the set on which this convergence holds and let $\omega \in \hat{\Omega}$. Hence, for each $\epsilon > 0$ there exists m_ϵ such that for $m \ge m_\epsilon$ we have $\|\Pi^{K_m}_n J - J\|_{\pi_n}(\omega) \le \epsilon$. Now, it is enough to remark that the projection can only get better when more random basis functions are used, since the space on which it is projected gets larger, implying that for $K \leq \tilde{K}$,

$$\|\Pi_n^K J - J\|_{\pi_n}(\omega) \geq \|\Pi_n^{\tilde{K}} J - J\|_{\pi_n}(\omega).$$

Therefore, also the original sequence converges at this ω, since given $\epsilon > 0$ for all $K \geq K_{m_\epsilon}$, we have

$$\|\Pi_n^K J - J\|_{\pi_n}(\omega) \leq \|\Pi_n^{K_{m_\epsilon}} J - J\|_{\pi_n}(\omega) \leq \epsilon.$$

□

Theorem A.1 holds equivalently if neural networks with more than 1 hidden layer are used. The proof is a straightforward extension of the proof given above.

A.4. **Convergence results.** The price of the Bermudan approximation of the American option can be expressed with the exact algorithm as

$$U_0 := \max\left(g(X_0), Q_0(X_0)\right),$$

and the price computed with the idealized algorithm is

$$U_0^K := \max\left(g(X_0), \tilde{Q}_0^K(X_0)\right)$$

while the price computed with the implementable algorithm is

$$U_0^{K,m} := \max\left(g(X_0), \frac{1}{m}\sum_{i=1}^m \hat{\tilde{Q}}_0^K(x_0, x_1^i, \dots, x_N^i)\right).$$

We provide two different convergence results with two different assumptions. The first result is based on [20] and needs a technical assumption that might not be satisfied in general. The second result is based on [70] and replaces this assumption by a stronger integrability assumption on the payoff.

A.4.1. *Convergence results based on* [20]. Combining the following two results, convergence of U_0^{K,m_K} to U_0 as $K \to \infty$ can be established by choosing a suitable sequence $(m_K)_{K\geq 1}$, under the assumption that $g(X_n)$ is square integrable for all $0 \leq n \leq N$.

Theorem A.4. *The idealized price U_0^K converges to the correct price U_0 $\tilde{\mathbb{P}}$-a.s. as $K \to \infty$.*

Theorem A.5. *We assume that $\mathbb{Q}[\alpha \Pi_n^K E_n P_{n+1}^K(X_n) = g(X_n)] = 0$ for all $0 \leq n \leq N-1$. Then, the implementable price $U_0^{K,m}$ converges almost surely to the idealized price U_0^K as $m \to \infty$.*

The proofs are a direct consequence of [20].

Proof of Theorems A.4 and A.5. The proofs are implied by the results presented in [20, Section 3]. We only need to establish that their assumption A_1 is satisfied. The assumption A_2 is actually not needed, as explained below.

Assumption A_1 is that $(\phi_k(X_n))_{k\geq 1}$ is total in $L^2(\sigma(X_n))$ for every $1 \leq n \leq N-1$, which is used to show that $\|\Pi_n^K Q_n - Q_n\|_{\pi_n}$ converges to 0. We replace this assumption by our Theorem A.1, which therefore yields $\tilde{\mathbb{P}}$-almost sure convergence in the result.

Assumption A_2 is that for every $1 \leq n \leq N$ and every $K > 0$, if $\sum_{k=0}^K \lambda_k \phi_k(X_n) = 0$ almost surely, then all $\lambda_k = 0$. This assumption is actually only needed for the projection weights to be uniquely defined, such that they can be expressed by the closed-form ordinary least squares formula. Otherwise, if this assumption is not satisfied, there exist several weight vectors θ, which all define the same projection $\Phi_K\theta$ minimizing the projection objective. By Gram–Schmidt, we can generate an orthonormal basis $(\tilde{\phi}_k)_{1\leq k\leq \tilde{K}(K)}$ of the linear subspace of L^2 that is spanned by $(\phi_k)_{1\leq k\leq K}$, with $\tilde{K}(K)\leq K$. By its definition, $(\tilde{\phi}_k)_{1\leq k\leq \tilde{K}(K)}$ satisfies assumption A_2; therefore, the results of [20, Section 3] can be applied. Finally, we note that the projections are the same, no matter whether $(\tilde{\phi}_k)_{1\leq k\leq \tilde{K}}$ or $(\phi_k)_{1\leq k\leq K}$ are used to describe the space that is spanned. We are interested in the convergence of the price. Considering the definition (14), we see that the price depends only on the projection but not on the used weights. Therefore, we can conclude that the same statements hold with our originally defined random basis functions $(\phi_k)_{1\leq k\leq K}$. □

The technical assumption that $\mathbb{Q}[\alpha\Pi_n^K E_n P_{n+1}^K(X_n)=g(X_n)]=0$ for all $0\leq n\leq N-1$ of the result of [20] that shows up in Theorem A.5 is not always satisfied. In particular, it is easy to construct examples of finite probability spaces, where this is not the case. Indeed, consider the easiest possible case of probability space which is a singleton, with a (deterministic) constant stock price without discounting, and then $\mathbb{Q}[\Pi_n^K E_n P_{n+1}^K(X_n)=g(X_n)]=1$. Therefore, in the next section, we provide a different proof based on the work of [70], which replaces this assumption with a slightly stronger integrability assumption on the payoff process.

A.4.2. *Convergence results based on* [70]. After the work of [20], improved theoretical guarantees to the original least squares Monte Carlo algorithm (LSM) have been proposed, such as [60, 23, 34]. An important improvement of the convergence results is done in [67, 68, 69, 70]. In particular, [67] raised the issue that [20] has additional restrictions on the law of the underlying Markov process such as the assumption in Theorem A.5 mentioned above. [67] proposed a generalized LSM algorithm and provides a proof of convergence in probability [67, Theorem 5.1]. In this theorem, the condition of [20] is not needed, but instead the payoff needs to be bounded almost surely [67, Definition 5.1 and 5.2]. [68] provides error estimates (convergence rates), even when the underlying process and payoff process are not necessarily in L^∞. Later, [69] provides a convergence result [69, Corollary 5.5] without the assumption in Theorem A.5 of [20], but with a bounded payoff process. However, this time, almost sure convergence is shown instead of convergence in probability. Finally, in the last paper [70], the assumption of having a bounded payoff process is replaced by a condition on its moments [70, Corollary 1]. We use this last result to prove our second convergence theorem. To state this result we define the truncation operator $\mathcal{T}_\lambda$ for truncation level $\lambda>0$ acting on any real-valued function f by

$$\mathcal{T}_\lambda f(x)=\begin{cases} f(x), & \text{if } |f(x)|\leq\lambda,\\ \lambda\,\mathrm{sign}(f(x)), & \text{otherwise.}\end{cases}$$

Theorem A.6. *Assume that there exists some* $2<p\leq\infty$ *such that*

$$M_p:=\max_{1\leq n\leq N}\|g(X_n)\|_{L^p}^p<\infty$$

and that all payoffs are non-negative. Moreover, assume that we use the truncated versions of the payoffs $g(X_n)$ *in Algorithm 1 as well as the truncated versions of the randomized neural networks, with truncation level* $1\leq\lambda<\infty$. *Then,*

$$\mathbb{E}\left[\left|U_0^{K,m}-U_0\right|\right]\xrightarrow[K,m\to\infty]{\tilde{\mathbb{P}}-a.s.}0,$$

when choosing $\lambda = m^{1/8}$.

The proof is a direct consequence of [70, Corollary 1].

Proof. Let us fix the number of paths m and the number of random basis functions K. Then, [70, Corollary 1] implies that

$$\begin{aligned}
&\mathbb{E}\left[\left|U_0^{K,m} - U_0\right|\right] \\
&\leq 6^N \left(\frac{C\lambda^2 \left(\sqrt{\nu c_0} \log^{\frac{1}{2}}(m) + \log^{\frac{1}{2}}(C_0)\right)}{\sqrt{m}} \right. \\
&\qquad \left. + 4\sqrt{\varepsilon} + \max_{n=1,\dots,N-1} \left(\inf_{f \in \mathcal{B}_n^{K,\lambda}} \|f - Q_n\|_{\pi_n} \right) + \left(\frac{8M_p \lambda^{(2-p)}}{p-2} \right)^{1/2} \right),
\end{aligned}$$

where $C_0 = C(c_0\nu+1)^4(C\lambda^4)^{2\nu(1+c_0)}$, $c_0 = 2(N+1)\log_2(e(N+1))$, C is a numerical constant with $1 \leq C < \infty$, and $\varepsilon \geq 0$ is defined in [70, Equation 13]. Here, ν is the Vapnik–Chervonenkis (VC) dimension of the set of randomized neural networks, which is finite according to [70, Remark 8]. For each exercise time $1 \leq n \leq N-1$ the set $\mathcal{B}_n^{K,\lambda}$ is defined to be the set of all λ-truncated randomized neural networks using the first K random basis functions (i.e., any truncated version of a linear combination of the basis functions $(\phi_k)_{1\leq k \leq K}$). In particular,

$$\mathcal{B}_n^{K,\lambda} = \{\mathcal{T}_\lambda f | f \in \operatorname{span}\{\phi_1, \dots, \phi_K\}\},$$

where $\mathcal{T}_\lambda$ is the operator truncating a function at λ. Note that for any function f, we have that

$$\|(\mathcal{T}_\lambda f - Q_n)1_{\{|Q_n|<\lambda\}}\|_{\pi_n} \leq \|(f - Q_n)1_{\{|Q_n|<\lambda\}}\|_{\pi_n}.$$

Therefore,

$$\begin{aligned}
&\inf_{f \in \mathcal{B}_n^{K,\lambda}} \|(f - Q_n)1_{\{|Q_n|<\lambda\}}\|_{\pi_n} \\
&\leq \inf_{f \in \operatorname{span}\{\phi_1,\dots,\phi_K\}} \|(f - Q_n)1_{\{|Q_n|<\lambda\}}\|_{\pi_n} \\
&\leq \inf_{f \in \operatorname{span}\{\phi_1,\dots,\phi_K\}} \|f - Q_n\|_{\pi_n} = \|\Pi_n^K Q_n - Q_n\|_{\pi_n}.
\end{aligned}$$

Hence, we can now bound the approximation error with truncated randomized neural networks by

$$\begin{aligned}
&\inf_{f \in \mathcal{B}_n^{K,\lambda}} \|f - Q_n\|_{\pi_n} \\
&\leq \inf_{f \in \mathcal{B}_n^{K,\lambda}} \left(\|(f - Q_n)1_{\{|Q_n|<\lambda\}}\|_{\pi_n} + \|(f - Q_n)1_{\{|Q_n|\geq\lambda\}}\|_{\pi_n} \right) \\
&\leq \inf_{f \in \mathcal{B}_n^{K,\lambda}} \|(f - Q_n)1_{\{|Q_n|<\lambda\}}\|_{\pi_n} + \sup_{f \in \mathcal{B}_n^{K,\lambda}} \|(f - Q_n)1_{\{|Q_n|\geq\lambda\}}\|_{\pi_n} \\
&\leq \|\Pi_n^K Q_n - Q_n\|_{\pi_n} + 2\|Q_n 1_{\{|Q_n|\geq\lambda\}}\|_{\pi_n},
\end{aligned}$$

where in the last inequality we used that functions in $\mathcal{B}_n^{K,\lambda}$ are truncated at λ, implying that they are bounded by $|Q_n|$ on the set $\{|Q_n| \geq \lambda\}$. Moreover, we can choose $\varepsilon = 1/m$, replace $\lambda = m^{1/8}$ and simplify all expressions by using one common constant $\tilde{C}$ to rewrite

$$\begin{aligned}
&\mathbb{E}\left[\left|U_0^{K,m} - U_0\right|\right] \\
&\le \tilde{C}\Bigg(\frac{\log^{\frac{1}{2}}(m)}{m^{1/4}} + \frac{1}{\sqrt{m}} \\
&\quad + \max_{n=1,\dots,N-1}\left(\|\Pi_n^K Q_n - Q_n\|_{\pi_n} + 2\|Q_n 1_{\{|Q_n|\ge m^{1/8}\}}\|_{\pi_n}\right) + m^{\frac{2-p}{16}}\Bigg).
\end{aligned}$$

Now, it suffices to note that the terms $\|\Pi_n^K Q_n - Q_n\|_{\pi_n}$ converge to 0 as $K \to \infty$ by Theorem A.1, the terms $\|Q_n 1_{\{|Q_n|\ge m^{1/8}\}}\|_{\pi_n}$ converge to 0 as $m \to \infty$ by dominated convergence, and the remaining terms trivially converge to 0 as $m \to \infty$. □

Appendix B. Convergence of the randomized fitted Q-iteration (RFQI).

Similarly as in Appendix A, we first introduce some additional technical notation needed for the proofs. Then, we describe the steps from the theoretical idea of RFQI to its implementable version that was presented in Section 3. In contrast to Appendix A, the algorithms described here are applied simultaneously for all times. Again, the proof is a direct consequence of [64] and Theorem A.1, but it is given in detail for completeness.

B.1. **Definitions.** In Section 6, [64] introduced a reinforcement learning version of their optimal stopping algorithm, where a stopping function is learned that generalizes over time. In particular, instead of learning a different function for each time step, a single function that gets the time as input is learned with an iterative scheme. In accordance with this, the random basis functions are redefined such that they also take time as input.

$$\begin{aligned}
&\phi_k : \mathbb{R}^d \times \{0,\dots,N-1\} \to \mathbb{R}, \\
&(x,n) \mapsto \phi_k(x,n) := \sigma(\alpha_k^\top (x,n)^\top + \beta_k),
\end{aligned}$$

with $\alpha_k \in \mathbb{R}^{d+1}$ and $\beta_k \in \mathbb{R}$. For $0 \le n \le N-1$ let $\Phi_{K,n}$ be defined similarly to before as

$$(\Phi_{K,n}\theta)(x) := \theta^\top \phi(x,n) := \sum_{k=1}^K \theta_k \phi_k(x,n),$$

for $\theta \in \mathbb{R}^K$ and $x \in \mathbb{R}^d$. Moreover, let $\Phi_k := (\Phi_{K,0},\dots,\Phi_{K,N-1})$, such that

$$\Phi_k\theta := (\Phi_{K,0}\theta,\dots,\Phi_{K,N-1}\theta).$$

In the following, we consider the product space $(L^2)^N := L^2(\pi_0) \times \cdots \times L^2(\pi_{N-1})$, which is the space on which the functions for all time steps can be defined concurrently. For $J = (J_0,\dots,J_{N-1}) \in (L^2)^N$ we define the norm

$$\|J\|_\pi := \frac{1}{N}\sum_{n=0}^{N-1}\|J_n\|_{\pi_n},$$

where $\|\cdot\|_{\pi_n}$ is as defined in Appendix A. Let us define the projection operator Π^K as

$$(\Pi^K J) := \arg\min_{\Phi_K\theta}\|\Phi_K\theta - J\|_\pi,$$

for $J = (J_0, \dots, J_{N-1}) \in (L^2)^N$. Finally, we define the operator

$$H : (L^2)^N \to (L^2)^N, \quad \begin{pmatrix} J_0 \\ \vdots \\ J_{N-2} \\ J_{N-1} \end{pmatrix} \mapsto \begin{pmatrix} \alpha E_0 \max(g, J_1) \\ \vdots \\ \alpha E_{N-2} \max(g, J_{N-1}) \\ \alpha E_{N-1} g \end{pmatrix}, \tag{15}$$

where E_n and g are as defined previously.

B.2. **Theoretical description of the algorithm.** Based on the definitions in Appendix A.2, we first introduce the *exact algorithm* and then give the two-step approximation with the *idealized* and *implementable algorithm.*

B.2.1. *Exact algorithm.* Let Q_n be as defined in (12), and then $Q := (Q_0, \dots, Q_{N-1})$ satisfies $Q = HQ$ by definition. In particular, Q is a fixed point of H. It was shown in [64, Section 6] that H is a contraction with respect to the norm $\|\cdot\|_\pi$ with contraction factor α. Hence, the Banach fixed point theorem implies that there exists a unique fixed point, which therefore has to be Q, and that for any starting element $J^0 \in (L^2)^N$, J^i converges to Q in $(L^2)^N$ as $i \to \infty$, where $J^{i+1} := HJ^i$. This yields a way to find the *exact algorithm* Q iteratively.

B.2.2. *Idealized algorithm.* The combined operator $\Pi^K H$ is a contraction on the space $\Pi^K (L^2)^N$, since the projection operator is a non-expansion as outlined in [64, Section 6]. The *idealized algorithm* is then defined as the unique fixed point $\tilde{Q}^K$ of $\Pi^K H$, which can again be found by iteratively applying this operator to an arbitrary starting point. Since any element in $\Pi^K (L^2)^N$ is given as $\Phi_K \theta$ for some weight vector $\theta \in \mathbb{R}^K$, this iteration can equivalently be given as iteration on the weight vectors. To do this, let us assume without loss of generality that $(\phi_k)_{1 \le k \le K}$ are linearly independent (if not, see the strategy in Proof of Theorem A.4 and A.5). Then, given some starting weight vector θ_K^0, the iterative application of $\Pi^K H$ defines the weight vectors

$$\begin{aligned} \theta_K^{i+1} :=& \alpha \left(\mathbb{E}\left[\sum_{n=0}^{N-1} \phi_{1:K}^\top (X_n, n) \phi_{1:K}(X_n, n) \right] \right)^{-1} \\ & \times \mathbb{E}\left[\sum_{n=0}^{N-1} \phi_{1:K}^\top (X_n, n) \cdot \max\left(g(X_{n+1}), (\Phi_{K,n+1}\theta_K^i)(X_{n+1}) \right) \right], \end{aligned}$$

where $\phi_{1:K} = (\phi_1, \dots, \phi_K)$. This closed-form solution is exactly the ordinary least squares (OLS) formula, and this result was shown in [64, Section 6].

B.2.3. *Implementable algorithm.* An *implementable version* of this iteration is defined by the Monte Carlo approximation of the weight vectors. In particular, we assume that m realizations $(x_0^j, \dots, x_N^j)_{1 \le j \le m}$ of X are sampled and fixed for all iterations. Then, for $\hat{\theta}_{K,m}^0 = \theta_K^0$ we iteratively define

$$\hat{\theta}_{K,m}^{i+1} := \alpha \left(\sum_{j=1}^{m} \sum_{n=0}^{N-1} \phi_{1:K}^\top (x_n^j, n) \phi_{1:K}(x_n^j, n) \right)^{-1}$$

$$\times \sum_{j=1}^{m}\sum_{n=0}^{N-1} \phi_{1:K}^\top(x_n^j, n) \cdot \max\left(g(x_{n+1}^j), (\Phi_{K,n+1}\hat\theta_{K,m}^i)(x_{n+1}^j)\right),$$

which in turn defines $\hat{Q}^{K,m,i} := \Phi_K \hat\theta^i_{K,m}$. As explained in [64, Section 6], this implementable iteration can equivalently be described as iteratively applying the operator $\widehat{\Pi^K H}$. Here $\widehat{\Pi^K H}$ is identical to $\Pi^K H$, but with the measures π_n replaced by the empirical measures $\hat\pi_n$ arising from the sampled trajectories $(x_0^j, \dots, x_N^j)_{1\le j\le m}$. Hence, $\widehat{\Pi^K H}$ is also a contraction, and Banach's fixed point theorem implies convergence to the unique fixed point

$$\hat{Q}^{K,m,i} \xrightarrow{i\to\infty} \hat{Q}^{K,m} =: \Phi_K \hat\theta^\star_{K,m}.$$

We note that this also implies that $\hat\theta^i_{K,m} \xrightarrow{i\to\infty} \hat\theta^\star_{K,m}$.

B.3. **Convergence result.** In the following, we show that prices of Bermudan options computed with the two approximation steps of the exact algorithm converge to the correct price, as $K, m \to \infty$. The prices are defined similarly as in Appendix A.4. Hence, it is enough to show that $\hat{Q}^{K,m_i,i}$ converges to $\tilde{Q}^K$ as $i \to \infty$ and that $\tilde{Q}^K$ converges to Q as $K \to \infty$.

Theorem B.1. *$\tilde{Q}^K$ converges $\tilde{\mathbb{P}}$-a.s. to Q as $K \to \infty$, i.e.,*

$$\|\tilde{Q}^K - Q\|_\pi \xrightarrow[K\to\infty]{\tilde{\mathbb{P}}-a.s.} 0.$$

Proof. First, let us recall [64, Theorem 3], which states that for $0 < \kappa < 1$, the contraction factor of $\Pi^K H$, we have

$$\|\tilde{Q}^K - Q\|_\pi \le \frac{1}{\sqrt{1-\kappa^2}} \|\Pi^K Q - Q\|_\pi.$$

Now, since Π^K is a non-expansion, and H a is contraction with factor α, we have $\kappa \le \alpha < 1$. Therefore, for every K we have

$$\|\tilde{Q}^K - Q\|_\pi \le \frac{1}{\sqrt{1-\alpha^2}} \|\Pi^K Q - Q\|_\pi. \tag{16}$$

Finally, we remark that Theorem A.1 holds equivalently for the norm $\|\cdot\|_\pi$, since the universal approximation theorem can equivalently be applied to the functions with the combined input (x, n). Hence, the right hand side of (16) converges to 0 $\tilde{\mathbb{P}}$-a.s. as $K \to \infty$. □

We recall that the weight vectors $\hat\theta^i_{K,m}$ are random variables since they depend on the m sampled trajectories of X.

Lemma B.2. *For any fixed $i \in \mathbb{N}$ we have that $\hat\theta^i_{K,m}$ converges to θ^i_K $\mathbb{Q}$-a.s. as $m \to \infty$.*

Proof. The proof follows the proof of [64, Theorem 2]. We introduce the intermediate weight as

$$\tilde\theta^i_{K,m} := \alpha \left(\sum_{j=1}^{m}\sum_{n=0}^{N-1} \phi_{1:K}^\top(x_n^j, n)\phi_{1:K}(x_n^j, n) \right)^{-1}$$

$$\times \sum_{j=1}^{m} \sum_{n=0}^{N-1} \phi_{1:K}^{\top}(x_n^j, n) \cdot \max\left(g(x_{n+1}^j), (\Phi_{K,n+1}\theta_K^{i-1})(x_{n+1}^j) \right).$$

Then, it is clear that $\tilde{\theta}_{K,m}^i$ converges to θ_K^i $\mathbb{Q}$-a.s. as $m \to \infty$, by the strong law of large numbers. Hence, $\delta_i(m) := |\tilde{\theta}_{K,m}^i - \theta_K^i|_2$ converges to 0 $\mathbb{Q}$-a.s. Moreover, for suitably chosen random variables $A_i(m)$ that remain bounded as $m \to \infty$, we have

$$\hat{\theta}_{K,m}^i - \tilde{\theta}_{K,m}^i = A_i(m)|\hat{\theta}_{K,m}^{i-1} - \theta_K^{i-1}|_2.$$

Therefore, we have by the triangle inequality

$$|\hat{\theta}_{K,m}^i - \theta_K^i|_2 \leq \delta_i(m) + A_i(m)|\hat{\theta}_{K,m}^{i-1} - \theta_K^{i-1}|_2.$$

Since (by our choice) we start with the same weight vector $\hat{\theta}_{K,m}^0 = \theta_K^0$, we can conclude by induction that

$$|\hat{\theta}_{K,m}^i - \theta_K^i|_2 \xrightarrow[m\to\infty]{\mathbb{Q}-a.s.} 0.$$

However, we remark that this proof works only as long as i is fixed, but not in the limit $i \to \infty$, because the inductive steps would lead to an infinite sum. □

Theorem B.3. *Let $K \in \mathbb{N}$ be fixed. Then, there exists a random sequence $(m_i)_{i\geq 0}$ such that $\hat{Q}^{K,m_i,i}$ converges $\mathbb{Q}$-a.s. to $\tilde{Q}^K$ as $i \to \infty$, i.e.,*

$$\|\hat{Q}^{K,m_i,i} - \tilde{Q}^K\|_\pi \xrightarrow[i\to\infty]{\mathbb{Q}-a.s.} 0.$$

Proof. Let us define $\theta_K^\star \in \mathbb{R}^K$ to be the weight vector of the unique fixed point $\tilde{Q}^K$ of $\Pi^K H$, i.e., $\tilde{Q}^K = \Phi_K \theta_K^\star$. From Banach's fixed point theorem we know that $|\theta_K^i - \theta_K^\star|_2 \to 0$ as $i \to \infty$.

With Lemma B.2 we know that for every $i \in \mathbb{N}$ there exists $\Omega_i \subset \Omega$ with $\mathbb{Q}(\Omega_i) = 1$ such that $\hat{\theta}_{K,m}^i(\omega)$ converges to θ_K^i for all $\omega \in \Omega_i$. Let $\Omega_\infty := \cap_{i=1}^\infty \Omega_i$ be the set on which this convergence holds for all $i \in \mathbb{N}$, and then $\mathbb{Q}(\Omega_\infty) = 1$. Fix $\omega \in \Omega_\infty$. Now, let us choose $m_0 = 0$; and for every $i > 0$, $m_i > m_{i-1}$ such that $|\hat{\theta}_{K,m_i}^i(\omega) - \theta_K^i|_2 \leq 1/i$. Therefore, we obtain that

$$|\hat{\theta}_{K,m_i}^i(\omega) - \theta_K^\star|_2 \leq |\hat{\theta}_{K,m_i}^i(\omega) - \theta_K^i|_2 + |\theta_K^i - \theta_K^\star|_2 \leq \frac{1}{i} + |\theta_K^i - \theta_K^\star|_2,$$

which converges to 0 when i tends to infinity. □

Appendix C. **Convergence of the randomized recurrent least squares Monte Carlo (RRLSM).** In this section, we extend the results of Appendix A to the non-Markovian setting, where we assume that the path up to the current time is a Markov process. In particular, given a discrete time stochastic process $X = (X_0, \ldots, X_N)$ as before, we assume that its extension $Z = (Z_0, \ldots, Z_N)$ with $Z_n = (X_n, X_{n-1}, \ldots, X_0, 0, \ldots, 0)$ taking values in $\mathbb{R}^{N+1\times d}$ for all $0 \leq n \leq N$ is a Markov process. Hence, all results of Appendix A hold similarly up to replacing X by Z, and they also hold for payoff functions that depend on the entire path of X up to the current time. In particular, this immediately implies that RLSM with the path input Z approximates the correct price of the Bermudan option arbitrarily well as $K \to \infty$. Therefore, it is only left to show that an equivalent result to Theorem A.1 holds for our randomized recurrent neural network (6), which takes X as input instead of Z but makes use of a latent variable in which information about the past is stored.

Fix some $1 \leq n \leq N-1$ and let π_n now be the distribution of Z_n under $\mathbb{Q}$. Moreover, let the basis functions $\phi^n = (\phi_k^n)_{k\geq 1}$ be now given by the n-th latent variable h_n of (6). In particular, we define ϕ_k^n as the function mapping $z_n = (x_n, x_{n-1}, \dots, x_0, 0, \dots, 0)$ to the k-th coordinate of the recursively defined vector

$$h_n = \sigma(A_x x_n + A_h h_{n-1} + b), \tag{17}$$

where $h_{-1} = 0$. By abuse of notation, for growing k we let the matrices grow by adding new rows of random elements to b, A_x and A_h and filling up the new columns of previous rows of A_h with zeros. Like this, ϕ_k^n is well defined for all $k \geq 1$. The operator Π_n^K is defined similarly as before, but with this new set of basis functions, defined on the set of π_n-integrable functions J. Then, we have to show that the following result is true, so that the assumptions for Theorem A.4 and A.5 are satisfied. The remainder of their proof works as before.

Proposition C.1. *If the activation function σ is invertible, then for all $0 \leq n \leq N-1$,*

$$\|\Pi_n^K Q_n - Q_n\|_{\pi_n} \xrightarrow[K\to\infty]{\tilde{\mathbb{P}}\text{-}a.s.} 0. \tag{18}$$

Before we start with the proof, we remark that standard results for the approximation of dynamical systems with RNNs [57] and reservoir computing systems [35] do not apply here, since the dynamical system to approximate $Q = (Q_0, \dots, Q_{N-1})$ is not time-invariant (in the language of [35]).

Proof. First, we note that it is enough to show that for any $\epsilon > 0$ there exists some size $K \in \mathbb{N}$ and weight matrices b, A_x, A_h such that the corresponding neural network approximation $\tilde{\Pi}_K^n Q_n$ satisfies $\|\tilde{\Pi}_n^K Q_n - Q_n\|_{\pi_n} < \epsilon$ for all $0 \leq n \leq N-1$. Indeed, if this is true, the convergence (18) follows by the same arguments as in Lemma A.3 and Theorem A.1.

Second, we note that it is enough to show the statement above for any fixed n separately, i.e., that for each $0 \leq n \leq N-1$ and $\epsilon > 0$ there exist $K^n \in \mathbb{N}$ and weight matrices b^n, A_x^n, A_h^n such that the corresponding neural network approximation $\tilde{\Pi}_K^n Q_n$ satisfies $\|\tilde{\Pi}_n^K Q_n - Q_n\|_{\pi_n} < \epsilon$. Indeed, if this is true, the stronger statement follows immediately by setting

$$A_x = \begin{pmatrix} A_x^0 \\ \vdots \\ A_x^{N-1} \end{pmatrix}, \quad A_h = \begin{pmatrix} A_h^0 & & \\ & \ddots & \\ & & A_h^{N-1} \end{pmatrix} \text{ and } b = \begin{pmatrix} b^0 \\ \vdots \\ b^{N-1} \end{pmatrix}.$$

Hence, let us fix some $\epsilon > 0$ and $0 \leq n \leq N-1$ and let us assume that $d = 1$ for simplicity of notation, while the extension to $d > 1$ is immediate. We know from the universal approximation theorem [41, Theorem 1] that there exists some neural network f such that $\|f - Q_n\|_{\pi_n} < \epsilon$. The difference between the approximation $\tilde{\Pi}_K^n$ and f is that $\tilde{\Pi}_K^n$ gets a recurrent input, while f gets the entire path as input. However, since n is fixed and finite, we can simply accumulate the same path information in h_n by setting $\hat{b} = 0$, $\hat{A}_x = (1, 0, \dots, 0)^\top \in \mathbb{R}^n$ and

$$\hat{A}_h = \begin{pmatrix} 0 & \dots & & & 0 \\ 1 & 0 & \dots & & 0 \\ 0 & \ddots & & & \vdots \\ \vdots & & & & \\ 0 & \dots & 0 & 1 & 0 \end{pmatrix} \in \mathbb{R}^{n \times n}.$$

Indeed, with this choice we have $\hat{h}_{n-1} = (\sigma(x_{n-1}), \sigma(\sigma(x_{n-2})), \dots, \sigma^{(n)}(x_0))^\top$ according to (17). It remains to show that the input z_n to f can be replaced by $(x_n, \hat{h}_{n-1})$. For this, let us define the function

$$\varphi : (x_n, \dots, x_0, 0, \dots, 0) \mapsto (x_n, \sigma(x_{n-1}), \dots, \sigma^{(n)}(x_0)).$$

Under the assumption that σ is invertible, φ also is, and there exists a function $\tilde{Q}_n$ such that $\tilde{Q}_n \circ \varphi = Q_n$. Since Q_n is integrable with respect to π_n, the change of variables formula implies that $\tilde{Q}_n$ is integrable with respect to $\varphi^{-1} \circ \pi_n$ and $\mathbb{E}^{(\varphi^{-1} \circ \pi_n)}[\tilde{Q}_n] = \mathbb{E}^{\pi_n}[\tilde{Q}_n \circ \varphi] = \mathbb{E}^{\pi_n}[Q_n]$. Therefore, there exists a neural network $\tilde{f} = \tilde{\beta}^\top \sigma(\tilde{A} \cdot + \tilde{b})$ such that

$$\|\tilde{f} \circ \varphi - Q_n\|_{\pi_n} = \|(\tilde{f} - \tilde{Q}_n) \circ \varphi\|_{\pi_n} = \|\tilde{f} - \tilde{Q}_n\|_{\varphi^{-1} \circ \pi_n} < \epsilon.$$

By extending $\hat{b}, \hat{A}_x, \hat{A}_h$ to

$$b = \begin{pmatrix} \hat{b} \\ \tilde{b} \end{pmatrix}, \quad A_x = \begin{pmatrix} \hat{A}_x \\ \tilde{A}_1 \end{pmatrix}, \quad A_h = \begin{pmatrix} \hat{A}_h & 0 \\ \tilde{A}_{2:n+1} & 0 \end{pmatrix},$$

where $\tilde{A} = (\tilde{A}_1, \tilde{A}_{2:n+1})$, we get

$$h_n = \begin{pmatrix} \hat{h}_n \\ \tilde{h}_n \end{pmatrix} = \begin{pmatrix} \sigma(\hat{A}_x x_n + \hat{A}_h \hat{h}_{n-1} + \hat{b}) \\ \sigma(\tilde{A} \varphi(z_n) + \tilde{b}) \end{pmatrix},$$

where $\tilde{\beta}^\top \tilde{h}_n = \tilde{f}(\varphi(z_n))$. Therefore, we can conclude the proof, since the corresponding approximation $\tilde{\Pi}_K^n$ satisfies $\|\tilde{\Pi}_K^n Q_n - Q_n\|_{\pi_n} \leq \|\tilde{f} \circ \varphi - Q_n\|_{\pi_n} \leq \epsilon$. □

Remark C.2. The idea of the proof is to use the recurrent structure only to recover the path-wise input z_n for which the standard feed-forward neural network approximation results can be used. This is clearly less efficient than using the path-wise input directly. However, in practice, the recurrent neural network approach is usually more efficient than the path-wise approach, finding better ways to store and process the past information than the one given in the proof. This is in line with our empirical findings.

C.1. **Stopping of a fractional Brownian motion – table.**

The results shown in the plots of Section 7.3 are given in Table 16.

Table 16. Results of stopping a fractional Brownian Motion for different Hurst parameters.

	price								duration							
H	DOS	pathDOS	RLSM	RRLSM	FQI	RFQI	RRFQI	pathRFQI	DOS	pathDOS	RLSM	RRLSM	FQI	RFQI	RRFQI	pathRFQI
0.01	0.85 (0.02)	1.48 (0.01)	0.84 (0.01)	1.45 (0.01)	0.79 (0.01)	0.78 (0.02)	0.85 (0.07)	1.09 (0.08)	1m15s	2m59s	0s	1s	9s	5s	18s	18s
0.05	0.67 (0.02)	1.24 (0.01)	0.65 (0.01)	1.24 (0.01)	0.68 (0.01)	0.67 (0.02)	0.71 (0.04)	0.99 (0.07)	1m15s	3m 1s	0s	1s	9s	4s	19s	19s
0.1	0.50 (0.02)	0.99 (0.01)	0.49 (0.01)	1.02 (0.01)	0.57 (0.01)	0.55 (0.01)	0.56 (0.05)	0.83 (0.03)	1m12s	2m58s	0s	1s	10s	4s	19s	20s
0.15	0.37 (0.02)	0.77 (0.01)	0.38 (0.01)	0.82 (0.01)	0.47 (0.02)	0.45 (0.02)	0.47 (0.05)	0.65 (0.03)	1m13s	2m59s	0s	1s	9s	4s	19s	18s
0.2	0.28 (0.01)	0.60 (0.01)	0.31 (0.01)	0.64 (0.01)	0.38 (0.01)	0.35 (0.09)	0.31 (0.02)	0.53 (0.02)	1m15s	2m58s	1s	1s	9s	4s	19s	17s
0.25	0.23 (0.01)	0.44 (0.01)	0.25 (0.01)	0.49 (0.01)	0.29 (0.01)	0.26 (0.05)	0.26 (0.04)	0.39 (0.02)	1m14s	2m58s	1s	1s	9s	4s	18s	19s
0.3	0.18 (0.01)	0.30 (0.01)	0.20 (0.01)	0.36 (0.01)	0.21 (0.01)	0.17 (0.01)	0.15 (0.01)	0.27 (0.01)	1m13s	2m57s	1s	1s	9s	4s	18s	18s
0.35	0.13 (0.01)	0.19 (0.01)	0.15 (0.01)	0.25 (0.01)	0.14 (0.01)	0.13 (0.02)	0.12 (0.03)	0.17 (0.01)	1m15s	2m57s	1s	1s	9s	4s	18s	19s
0.4	0.08 (0.01)	0.10 (0.01)	0.10 (0.01)	0.14 (0.01)	0.09 (0.01)	0.06 (0.01)	0.06 (0.01)	0.10 (0.02)	1m15s	2m59s	1s	1s	9s	4s	18s	19s
0.45	0.04 (0.01)	0.03 (0.01)	0.05 (0.01)	0.06 (0.01)	0.04 (0.01)	0.02 (0.01)	0.03 (0.01)	0.05 (0.01)	1m14s	2m58s	0s	1s	9s	4s	18s	18s
0.5	0.00 (0.00)	0.01 (0.01)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.01)	0.01 (0.01)	0.00 (0.01)	1m14s	2m57s	0s	1s	9s	4s	18s	18s
0.55	0.03 (0.01)	0.02 (0.01)	0.03 (0.01)	0.05 (0.01)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1m16s	3m 0s	1s	1s	9s	4s	17s	18s
0.6	0.07 (0.00)	0.09 (0.01)	0.08 (0.01)	0.10 (0.01)	0.00 (0.01)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)	1m12s	2m56s	1s	1s	9s	4s	17s	18s
0.65	0.10 (0.01)	0.14 (0.01)	0.12 (0.01)	0.16 (0.01)	0.00 (0.00)	0.01 (0.01)	0.00 (0.00)	0.00 (0.00)	1m13s	2m59s	1s	1s	9s	4s	17s	18s
0.7	0.14 (0.01)	0.19 (0.01)	0.16 (0.01)	0.20 (0.01)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1m13s	2m57s	1s	1s	9s	4s	18s	18s
0.75	0.18 (0.01)	0.23 (0.01)	0.19 (0.00)	0.23 (0.01)	0.00 (0.00)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)	1m15s	2m55s	1s	1s	9s	4s	18s	18s
0.8	0.22 (0.01)	0.26 (0.01)	0.23 (0.01)	0.26 (0.01)	0.00 (0.01)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)	1m15s	2m58s	1s	1s	9s	4s	17s	18s
0.85	0.26 (0.00)	0.29 (0.01)	0.27 (0.01)	0.29 (0.01)	0.00 (0.01)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)	1m16s	2m55s	1s	1s	9s	4s	18s	18s
0.9	0.30 (0.01)	0.33 (0.01)	0.30 (0.00)	0.32 (0.00)	0.00 (0.01)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1m14s	2m55s	1s	1s	9s	4s	18s	18s
0.95	0.34 (0.01)	0.35 (0.00)	0.34 (0.01)	0.35 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	1m 9s	2m55s	1s	1s	9s	4s	18s	18s
0.999	0.38 (0.01)	0.39 (0.01)	0.38 (0.01)	0.38 (0.00)	0.00 (0.00)	0.01 (0.01)	0.00 (0.00)	0.00 (0.00)	1m18s	2m45s	1s	1s	9s	4s	18s	18s

C.2. **Non-Markovian stock models – additional tables.** Additional results for the non-Markovian setting of a Heston model without the variance as input are given in Tables 17-19.

TABLE 17. Max call option on Heston for different numbers of stocks d.

	price							duration						
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP	LSM	DOS	NLSM	RLSM	FQI	RFQI	EOP
5	8.34 (0.07)	8.29 (0.09)	8.17 (0.06)	8.31 (0.07)	8.23 (0.04)	8.34 (0.08)	8.23 (0.04)	11s	7s	3s	0s	3s	0s	0s
10	11.83 (0.07)	11.81 (0.09)	11.39 (0.16)	11.83 (0.07)	11.77 (0.04)	11.82 (0.05)	11.79 (0.05)	29s	6s	3s	0s	6s	0s	0s
50	19.60 (0.07)	20.04 (0.04)	18.14 (0.37)	19.32 (0.05)	20.05 (0.06)	20.08 (0.06)	20.06 (0.03)	8m50s	7s	3s	0s	6m36s	1s	0s
100	20.51 (0.09)	23.57 (0.07)	21.29 (0.46)	22.87 (0.04)	23.56 (0.07)	23.67 (0.05)	23.67 (0.05)	40m44s	9s	3s	0s	1h21m35s	1s	0s
500	-	31.62 (0.06)	28.38 (0.55)	31.33 (0.04)	-	32.09 (0.06)	32.14 (0.02)	-	44s	8s	1s	-	1s	0s
1000	-	34.99 (0.08)	33.03 (0.50)	35.06 (0.04)	-	35.83 (0.05)	35.84 (0.03)	-	1m16s	15s	2s	-	1s	0s
2000	-	37.77 (0.07)	36.77 (0.32)	38.83 (0.06)	-	39.64 (0.07)	39.61 (0.04)	-	2m17s	25s	4s	-	2s	0s

TABLE 18. Min put option on Heston for different numbers of stocks d and varying initial stock price x_0. Here, $r = 2\%$ is used as interest rate.

	price						duration					
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI	RFQI
5	12.29 (0.07)	12.26 (0.06)	12.12 (0.08)	12.25 (0.07)	12.38 (0.08)	12.34 (0.07)	12s	6s	3s	0s	2s	0s
10	16.55 (0.06)	16.54 (0.10)	16.03 (0.19)	16.50 (0.06)	16.63 (0.09)	16.64 (0.06)	30s	6s	3s	0s	10s	0s
50	25.24 (0.07)	25.66 (0.07)	23.67 (0.35)	24.87 (0.04)	25.71 (0.07)	25.68 (0.04)	8m42s	8s	3s	0s	7m34s	1s
100	26.84 (0.09)	29.22 (0.07)	26.47 (0.62)	28.45 (0.03)	29.26 (0.06)	29.32 (0.07)	42m26s	12s	4s	0s	1h24m 4s	1s
500	-	36.47 (0.05)	33.80 (0.65)	36.26 (0.05)	-	36.93 (0.04)	-	56s	13s	1s	-	1s
1000	-	39.25 (0.04)	37.01 (0.34)	39.33 (0.02)	-	39.93 (0.04)	-	1m49s	23s	2s	-	2s
2000	-	41.45 (0.03)	39.92 (0.26)	42.25 (0.05)	-	42.78 (0.04)	-	3m58s	43s	5s	-	2s

TABLE 19. Max call option on Heston for different numbers of stocks d. Here, $r = 5\%$ is used as interest rate, and $\delta = 10\%$ is used as dividend rate.

	price						duration					
d	LSM	DOS	NLSM	RLSM	FQI	RFQI	LSM	DOS	NLSM	RLSM	FQI	RFQI
5	4.82 (0.03)	4.78 (0.04)	4.68 (0.04)	4.75 (0.04)	4.29 (0.12)	4.57 (0.06)	12s	5s	3s	0s	2s	0s
10	7.20 (0.06)	7.16 (0.04)	6.92 (0.06)	7.13 (0.05)	6.60 (0.14)	6.76 (0.16)	29s	6s	3s	0s	8s	0s
50	13.48 (0.05)	13.98 (0.03)	12.44 (0.18)	13.69 (0.04)	13.79 (0.03)	13.72 (0.07)	8m34s	8s	3s	0s	7m 7s	1s
100	14.63 (0.07)	17.13 (0.06)	15.19 (0.32)	16.83 (0.04)	16.97 (0.07)	16.99 (0.04)	39m49s	12s	6s	0s	1h23m 4s	1s
500	-	24.31 (0.08)	21.83 (0.63)	24.37 (0.04)	-	24.69 (0.05)	-	54s	12s	1s	-	1s
1000	-	27.42 (0.07)	25.64 (0.55)	27.73 (0.03)	-	28.08 (0.06)	-	1m39s	23s	2s	-	2s
2000	-	30.10 (0.08)	29.27 (0.36)	31.09 (0.04)	-	31.50 (0.06)	-	3m47s	43s	5s	-	2s

Acknowledgments. The authors would like to thank Sebastian Becker, Patrick Cheredito, Blanka Horvath, Arnulf Jentzen, Hartmut Maennel and Louis Paulot for helpful feedback and discussions. In addition, the authors would like to warmly thank the quant team of Crédit Agricole CIB, and in particular Arthur Semin, Ryan Kurniawan and Wail El Allali for the great collaboration, which considerably improved the paper. Thanks to this collaboration, we provide the computation of the Greeks, we improved the sensitivity to the randomness of the hidden layers of

RLSM, and we improved the proof of convergence of RLSM. The authors would also like to thank the anonymous reviewers for their feedback leading to significant improvements of the paper. Moreover, the authors would like to acknowledge support for this project from the Swiss National Science Foundation (SNF grant 179114).

REFERENCES

[1] E. Abi Jaber and O. El Euch, Multifactor approximation of rough volatility models, *SIAM Journal on Financial Mathematics*, **10** (2019), 309-349.

[2] L. Andersen, A simple approach to the pricing of Bermudan swaptions in the multi-factor Libor Market model, *Mathematical Finance*, **3** (1999), 5-32.

[3] A. Bakan, Representation of measures with polynomial denseness in $L_p(\mathbb{R}, d\mu)$, $0 < p < \infty$, and its application to determinate moment problems, *Proceedings of the American Mathematical Society*, **136** (2008), 3579-3589.

[4] V. Bally and G. Pagès, A quantization algorithm for solving multi-dimensional discrete-time optimal stopping problems, *Bernoulli*, **9** (2003), 1003-1049.

[5] V. Bally, G. Pagès and J. Printems, A quantization tree method for pricing and hedging multidimensional American options, *Mathematical Finance*, **15** (2005), 119-168.

[6] P. Bank and D. Besslich, On Lenglart's theory of Meyer-sigma-fields and El Karoui's theory of optimal stopping, `arXiv:1810.08485`, Preprint, 2019.

[7] J. Barraquand and D. Martineau, Numerical valuation of high dimensional multivariate American securities, *The Journal of Financial and Quantitative Analysis*, **30** (1995), 383-405.

[8] C. Bayer, M. Eigel, L. Sallandt and P. Trunschke, Pricing high-dimensional Bermudan options with hierarchical tensor formats, *SIAM Journal on Financial Mathematics*, **14** (2023), 383-406.

[9] C. Bayer, P. Friz and J. Gatheral, Pricing under rough volatility, *Quantitative Finance*, **16** (2016), 887-904.

[10] S. Becker, P. Cheridito and A. Jentzen, Deep optimal stopping, *Journal of Machine Learning Research*, **20** (2019), Paper No. 74, 25 pp.

[11] S. Becker, P. Cheridito and A. Jentzen, Pricing and hedging American-style options with deep learning, *Journal of Risk and Financial Management*, **13** (2020), 158.

[12] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, 1996.

[13] B. Bouchard and X. Warin, Monte-Carlo valuation of American options: Facts and new algorithms to improve existing methods, in *Proceedings of the Numerical Methods in Finance: Bordeaux, June 2010*, Springer, 2012, 215-255.

[14] P. P. Boyle, A. W. Kolkiewicz and K. S. Tan, An improved simulation method for pricing high-dimensional American derivatives, *Mathematics and Computers in Simulation*, **62** (2003), 315-322.

[15] M. Broadie and P. Glasserman, A stochastic mesh method for pricing high-dimensional American options, *Journal of Computational Finance*, **7** (2004), 35-72.

[16] W. Cao, X. Wang, Z. Ming and J. Gao, A review on neural networks with random weights, *Neurocomputing*, **275** (2018), 278-287.

[17] J. F. Carriere, Valuation of the early-exercise price for options using simulations and non-parametric regression, *Insurance: Mathematics and Economics*, **19** (1996), 19-30.

[18] S. Chen, A. M. Devraj, A. Bušić and S. Meyn, Zap Q-learning for optimal stopping, in *Proceedings of the 2020 American Control Conference (ACC)*, IEEE, 2020, 3920-3925.

[19] E. Chevalier, S. Pulido and E. Zúñiga, American options in the Volterra Heston model, *SIAM Journal on Financial Mathematics*, **13** (2022), 426-458.

[20] E. Clément, D. Lamberton and P. Protter, *An Analysis of the Longstaff-Schwartz Algorithm for American Option Pricing*, Technical report, Cornell University Operations Research and Industrial Engineering, 2001.

[21] J. C. Cox, S. A. Ross and M. Rubinstein, Option pricing: A simplified approach, *Journal of Financial Economics*, **7** (1979), 229-263.

[22] M. de Bellefroid, *The Derivatives Academy*, 2022. `https://bookdown.org/maxime_debellefroid/MyBook/`.

[23] D. Egloff, Monte Carlo algorithms for optimal stopping and statistical learning, *The Annals of Applied Probability*, **15** (2005), 1396-1432.

[24] D. Egloff, M. Kohler and N. Todorovic, A dynamic look-ahead Monte Carlo algorithm for pricing Bermudan options, *The Annals of Applied Probability*, **17** (2007), 1138-1171.

[25] O. El Euch, M. Fukasawa and M. Rosenbaum, The microstructural foundations of leverage effect and rough volatility, *Finance and Stochastics*, **22** (2018), 241-280.

[26] O. El Euch, J. Gatheral and M. Rosenbaum, Roughening Heston, *Risk*, (2019), 84-89.

[27] O. El Euch and M. Rosenbaum, Perfect hedging in rough Heston models, *The Annals of Applied Probability*, **28** (2018), 3813-3856.

[28] N. El Karoui, Les aspects probabilistes du controle stochastique, in *Proceedings of the École d'été de Probabilités de Saint-Flour IX-1979*, Springer, 1981, 73-238.

[29] H. Föllmer and A. Schied, *Stochastic Finance: An Introduction in Discrete Time*, De Gruyter, 2016.

[30] C. Gallicchio, A. Micheli and L. Pedrelli, Deep reservoir computing: A critical experimental analysis, *Neurocomputing*, **268** (2017), 87-99.

[31] D. García, Convergence and biases of Monte Carlo estimates of American option prices using a parametric exercise rule, *Journal of Economic Dynamics and Control*, **27** (2003), 1855-1879.

[32] J. Gatheral, T. Jaisson and M. Rosenbaum, Volatility is rough, *Quantitative Finance*, **18** (2018), 933-949.

[33] J. Gatheral, P. Jusselin and M. Rosenbaum, The quadratic rough heston model and the joint S&P 500/VIX smile calibration problem, `arXiv:2001.01789`, Preprint, 2020.

[34] E. Gobet, J.-P. Lemor and X. Warin, A regression-based Monte Carlo method to solve backward stochastic differential equations, *The Annals of Applied Probability*, **15** (2005), 2172-2202.

[35] L. Gonon and J.-P. Ortega, Reservoir computing universality with stochastic inputs, *IEEE Transactions on Neural Networks and Learning Systems*, **31** (2020), 100-112.

[36] A. N. Gorban, I. Y. Tyukin, D. V. Prokhorov and K. I. Sofeikov, Approximation with random bases: Pro et contra, *Information Sciences*, **364** (2016), 129-145.

[37] H. Hanbali and D. Linders, American-type basket option pricing: A simple two-dimensional partial differential equation, *Quantitative Finance*, **19** (2019), 1689-1704.

[38] M. B. Haugh and L. Kogan, Pricing American options: A duality approach, *Operations Research*, **52** (2004), 258-270.

[39] C. Herrera, F. Krach and J. Teichmann, Estimating full Lipschitz constants of deep neural networks, `arXiv:2004.13135`, Preprint, 2020.

[40] S. L. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, *The Review of Financial Studies*, **6** (1993), 327-343.

[41] K. Hornik, Approximation capabilities of multilayer feedforward networks, *Neural Networks*, **4** (1991), 251-257.

[42] G.-B. Huang, L. Chen, C. K. Siew, et al., Universal approximation using incremental constructive feedforward networks with random Hidden nodes, *IEEE Transactions on Neural Networks*, **17** (2006), 879-892.

[43] S. Jain and C. W. Oosterlee, The stochastic grid bundling method: Efficient pricing of Bermudan options and their Greeks, *Applied Mathematics and Computation*, **269** (2015), 412-431.

[44] L. P. Kaelbling, M. L. Littman and A. W. Moore, Reinforcement learning: A survey, *Journal of Artificial Intelligence Research*, **4** (1996), 237-285.

[45] M. Kohler, A. Krzyżak and N. Todorovic, Pricing of high-dimensional American options by neural networks, *Mathematical Finance*, **20** (2010), 383-410.

[46] A. Kolodko and J. Schoenmakers, Iterative construction of the optimal Bermudan stopping time, *Finance and Stochastic*, **10** (2006), 27-49.

[47] B. Lapeyre and J. Lelong, Neural network regression for Bermudan option pricing, *Monte Carlo Methods and Applications*, **27** (2021), 227-247.

[48] P. Letourneau and L. Stentoft, Simulated Greeks for American options, *Quantitative Finance*, **23** (2023), 653-676.

[49] Y. Li, C. Szepesvari and D. Schuurmans, Learning exercise policies for American options, in *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, PMLR, 2009, 352-359.

[50] G. Livieri, S. Mouti, A. Pallavicini and M. Rosenbaum, Rough volatility: Evidence from option prices, *IISE Transactions*, **50** (2018), 767-776.

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

[52] M. Lukoševičius and H. Jaeger, Reservoir computing approaches to recurrent neural network training, *Computer Science Review*, **3** (2009), 127-149.

[53] G. Pagès, *Numerical Probability: An Introduction with Applications to Finance*, 1st edition, Springer, 2018.

[54] H. Pham, Optimal stopping, free boundary, and American option in a jump-diffusion model, *Applied Mathematics and Optimization*, **35** (1997), 145-164.

[55] L. C. G. Rogers, Monte Carlo valuation of American options, *Mathematical Finance*, **12** (2002), 271-286.

[56] L. C. G. Rogers, Dual valuation and hedging of Bermudan options, *SIAM Journal on Financial Mathematics*, **1** (2010), 604-608.

[57] A. M. Schäfer and H. G. Zimmermann, Recurrent neural networks are universal approximators, in *Proceedings of the 16th International Conference on Artificial Neural Networks–ICANN 2006: Athens, Greece*, Springer, 2006, 632-640.

[58] B. Schrauwen, D. Verstraeten and J. V. Campenhout, An overview of reservoir computing: Theory, applications and implementations, in *Proceedings of the 15th European Symposium on Artificial Neural Networks*, 2007, 471-482.

[59] M. Schweizer, On Bermudan options, in *Advances in Finance and Stochastics: Essays in Honour of Dieter Sondermann*, Springer, Berlin, Heidelberg, 2002, 257-270.

[60] L. Stentoft, Convergence of the least squares Monte Carlo approach to American option valuation, *Management Science*, **50** (2004), 1193-1203.

[61] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, MIT press, 2018.

[62] J. A. Tilley, Valuing American options in a path simulation model, *Insurance Mathematics and Economics*, **2** (1995), 169.

[63] J. N. Tsitsiklis and B. Van Roy, Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives, *IEEE Transactions on Automatic Control*, **44** (1999), 1840-1851.

[64] J. N. Tsitsiklis and B. Van Roy, Regression methods for pricing complex American-style options, *IEEE Transactions on Neural Networks*, **12** (2001), 694-703.

[65] D. Verstraeten, B. Schrauwen, M. D'Haene and D. Stroobandt, An experimental unification of reservoir computing methods, *Neural Networks*, **20** (2007), 391-403.

[66] H. Yu and D. P. Bertsekas, Q-learning algorithms for optimal stopping based on least squares, in *Proceedings of the 2007 European Control Conference (ECC)*, IEEE, 2007, 2368-2375.

[67] D. Z. Zanger, Convergence of a least-squares Monte Carlo algorithm for bounded approximating sets, *Applied Mathematical Finance*, **16** (2009), 123-150.

[68] D. Z. Zanger, Quantitative error estimates for a least-squares Monte Carlo algorithm for American option pricing, *Finance and Stochastics*, **17** (2013), 503-534.

[69] D. Z. Zanger, Convergence of a least-squares Monte Carlo algorithm for American option pricing with dependent sample data, *Mathematical Finance*, **28** (2018), 447-479.

[70] D. Z. Zanger, General error estimates for the Longstaff-Schwartz least-squares Monte Carlo algorithm, *Mathematics of Operations Research*, **45** (2020), 923-946.

[71] R. Zhang, Y. Lan, G.-B. Huang and Z.-B. Xu, Universal approximation of extreme learning machine with adaptive growth of Hidden nodes, *IEEE Transactions on Neural Networks and Learning Systems*, **23** (2012), 365-371.

Received April 2022; revised August 2023; early access December 2023.