To appear in *Quantitative Finance*, Vol. 00, No. 00, Month 20XX, 1–26

Pricing high-dimensional American options by kernel ridge regression

Wenbin Hu*† and Tomasz Zastawniak‡

†College of Economics, Hangzhou Dianzi University, Hangzhou, China
‡Department of Mathematics, University of York, Heslington, York YO10 5DD, United Kingdom

(*Received 00 Month 20XX; in final form 00 Month 20XX*)

In this paper, we propose using kernel ridge regression (KRR) to avoid the step of selecting basis functions for regression-based approaches in pricing high-dimensional American options by simulation. Our contribution is threefold. Firstly, we systematically introduce the main idea and theory of KRR and apply it to American option pricing for the first time. Secondly, we show how to use KRR with the Gaussian kernel in the regression-later method and give the computationally efficient formulas for estimating the continuation values and the Greeks. Thirdly, we propose to accelerate and improve the accuracy of KRR by performing local regression based on the bundling technique. The numerical test results show that our method is robust and has both higher accuracy and efficiency than the Least Squares Monte Carlo method in pricing high-dimensional American options.

Keywords: High-dimensional American Option; Monte Carlo; Regression-based Method; Kernel Ridge Regression; Machine Learning.

JEL Classification: C63, G13

1. Introduction

The pricing of American options through Monte Carlo simulation is an active and evolving area of research. Among all kinds of proposed approaches, regression-based Monte Carlo methods (proposed by Carriere (1996) and further developed by Tsitsiklis and Van Roy (2001) and Longstaff and Schwartz (2001)) have been becoming popular. Depending on the procedure of generating basis functions, regression-based methods can be categorized into two types: regression-now and regression-later (Glasserman and Yu (2004)). Up to now, the Least Squares Monte Carlo (LSM) method (Longstaff and Schwartz (2001)), one of the typical representatives of regression-now methods, is still the most successful method for pricing American options by simulation and it has become the method of choice for practitioners. Please see Glasserman (2003) and Kohler (2010) for more detailed review.

One of the drawbacks of regression-based methods is that there is not an objective way of choosing the basis functions for regression, especially for the high-dimensional options. As a result, nonparametric regression, kernel-based regression (Han *et al.* (2009)), robust regression (Jonen (2011)) and so on have been proposed to overcome or alleviate this issue. Among these approaches, the kernel method is appealing. Kernel-based methods belong to the field of machine learning, with support vector machine (SVM) and kernel regression the typical representatives. Machine learning

*Corresponding author. Email: hwbgood@hdu.edu.cn

for quantitative finance, such as derivative pricing and hedging, has been attracting increasing attention (e.g., Spiegeleer *et al.* (2018)). Spiegeleer *et al.* (2018) use Gaussian process regression to directly fit the option prices, without applying the option pricing methodology. Specifically, the regressors are the option parameters such as the initial share prices, volatility and maturity, and the explained variable is the option price. It can be considered as a high-level application of machine learning methods in pricing. By contrast, we focus on a low-level application, in which we use kernel regression in one of the key steps for pricing American options by simulation. With kernel regression, the inner product of basis functions can be replaced with a kernel function, which can expand the original space to a space with higher or even infinite dimension. Kernel-based methods have already been applied to American options pricing (see Han *et al.* (2009)), with promising results. However, the theoretical and implementation aspects of the method presented in Han *et al.* (2009) need further clarification, and one may actually have a better choice of the kernel-based methods (e.g., the method we use in this paper). Furthermore, no pricing results for high-dimensional American options have been reported, though it is where the kernel method can show its advantages.

Another newly proposed American option pricing method, the stochastic grid bundling method (SGBM) (Jain and Oosterlee (2015)) is noticeable. It is a hybrid method of the stochastic mesh method (Broadie and Glasserman (1997)), LSM and the stratified along the payoff function method (Barraquand and Martineau (1995)). SGBM has lower variance than LSM and it can calculate the Greeks rapidly. However, one drawback is that one needs to choose the basis functions and calculate their conditional expectations in a closed form. As a result, it is only applicable to some particular processes, and the calculation of the conditional expectations of the basis functions is not trivial.

In this paper, we propose using kernel ridge regression (KRR) to price high-dimensional American options. We use a new kernel-based approach, which is different from the method of Han *et al.* (2009). Our method also utilizes the idea of bundling as SGBM but it conducts a totally different regression step. SGBM needs to calculate the conditional expectation of the mapping function or the basis functions, which would be difficult in some cases. Thanks to KRR, we perform the regression directly on the original high-dimensional state space and avoid the problem of selecting basis functions. Furthermore, the calculation of the conditional expectation in SGBM is transferred to the kernel function, making the method simpler and more general.

The paper is organized as follows. Section 2 gives an introduction to the problem of pricing American options by simulation, including the problem formulation and the detail of regression-based methods. Section 3 then presents the theoretic aspects of KRR. In Section 4, the implementations of KRR for both regression-now and regression-later methods, as well as the parameter tuning method are discussed. An algorithm is provided to summarize our method. Numerical test results and discussion are given in Section 5. Finally, we conclude in Section 6.

2. Pricing American options by simulation

2.1. *The problem of pricing American options*

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space with a time horizon $[0, T]$, where $\mathbb{P}$ is the risk neutral measure and $\mathcal{F} = \{\mathcal{F}_t | 0 \leq t \leq T\}$ is the filtration. Denote by r_t the risk-free interest rate, which is assumed to be non-stochastic in this paper. Then we can define the risk-less savings account process $B_t = \exp(\int_0^t r_s ds)$ and the discount factor $D_{s,t} = B_s/B_t$. The problem of pricing an American option is to find the optimal expected discounted payoff:

$$V_0(\mathbf{S}_0) = E\left[D_{0,\tau^*} h_{\tau^*}(\mathbf{S}_{\tau^*})\right] = \sup_{\tau \in \mathcal{T}} E\left[D_{0,\tau} h_\tau(\mathbf{S}_\tau)\right] \tag{1}$$

where $\mathbf{S}_t \in R^d$ is the price vector of the underlying assets, $h_t(\mathbf{x})$ is the payoff function for exercise at t, $\mathcal{T}$ is the set of all stopping times with respect to $\mathcal{F}$ and τ^* is the optimal stopping time. In

practice, the American option is approximated by a Bermudan option, which can only be exercised at discreet time points.

The optimal exercise strategy can be determined via a backward recursive process starting from the maturity date, i.e.,

$$V_T(\mathbf{S}_T) = h_T(\mathbf{S}_T). \tag{2}$$

Assume the underlying assets follow a Markovian process and the time is discretized into N time-steps, with $t_0 = 0$ and $t_N = T$. The continuation value of the option at t_i can be written as (we use i as t_i to lighten notation)

$$Q_i(\mathbf{S}_i) = E[D_{i,i+1}V_{i+1}(\mathbf{S}_{i+1})|\mathbf{S}_i],\ 0 \le i \le N-1. \tag{3}$$

The corresponding option value at t_i is given by

$$V_i(\mathbf{S}_i) = \max\{Q_i(\mathbf{S}_i), h_i(\mathbf{S}_i)\}. \tag{4}$$

Equations (2) and (4) form a dynamic programming principle (DPP) for solving the optimal stopping problem (1). Using the notations above, the optimal stopping time of the problem can be written as

$$\tau_0^* = \inf\{t \ge 0 : Q_t(\mathbf{S}_t) \le h_t(\mathbf{S}_t)\}, \tag{5}$$

and the DPP can equivalently be written as

$$\begin{cases} \tau_N^* = N \\ \tau_i^* = \begin{cases} i, & Q_i(\mathbf{S}_i) \ge h_i(\mathbf{S}_i) \\ \tau_{i+1}^*, & \text{otherwise} \end{cases}, \quad i = N-1, ..., 0. \end{cases} \tag{6}$$

No matter in what form, the key step of implementing the DPP is to estimate the continuation values. In this paper, we only focus on solving the DPPs discussed above by the Monte Carlo method, though other general methods such as the binomial tree approximation can also be applied to this problem. Under DPP (4) (e.g., Tsitsiklis and Van Roy (2001)), the value of the option estimated by simulation is given by

$$\widehat{V}_0 = \frac{1}{M}\sum_{m=1}^{M} \max\left\{\widehat{Q}_0^{(m)}\left(\mathbf{S}_0^{(m)}\right),\ h_0\left(\mathbf{S}_0^{(m)}\right)\right\}, \tag{7}$$

and under DPP (6) (e.g., Longstaff and Schwartz (2001)) it is

$$\widehat{V}_0 = \frac{1}{M}\sum_{m=1}^{M} D_{0,\tau_{0,m}^*} h_{\tau_{0,m}^*}\left(\mathbf{S}_{\tau_{0,m}^*}^{(m)}\right), \tag{8}$$

where M is the number of simulated paths, $\widehat{Q}_0^{(m)}$ is the estimated continuation value corresponding to the asset prices $\mathbf{S}_0^{(m)}$ on path m at time t_0, and $\tau_{0,m}^*$ is the realized optimal stopping time on path m. Note that DPP (6) is usually more accurate and robust, as the estimation of the continuation value is only used to make the decision and has limited influence on the estimated option price. DPP (4) tends to give upper biased pricing values due to the convexity of the max function. In this paper, we use DPP (6) unless otherwise specified.

2.2. *Regression-based Monte Carlo methods*

The basic idea of regression-based Monte Carlo methods is to approximately estimate the continuation value $Q_i(\mathbf{S}_i)$ ($0 \leq i \leq N-1$) at each time-step i by regression. There are basically two ways to obtain the estimations: regression-now and regression-later (Glasserman and Yu (2004)).

The regression-now method, which is more common, is to directly estimate $Q_i(\mathbf{S}_i)$ by regression on a limited number of basis functions at t_i. Assuming the continuation value function $Q_i(\mathbf{x})$ belongs to the L^2 space, the linear regression can be written as

$$\widehat{Q}_i(\mathbf{S}_i) := \boldsymbol{\phi}^\top(\mathbf{S}_i)\boldsymbol{\beta}_i = \sum_{j=1}^{J} \beta_{i,j}\phi_j(\mathbf{S}_i), \tag{9}$$

where

$$\boldsymbol{\phi}(\mathbf{x}) = (\phi_1(\mathbf{x}), \phi_2(\mathbf{x}), ..., \phi_J(\mathbf{x}))^\top,\ \mathbf{x} \in \mathbb{R}^d \tag{10}$$

are the selected J basis functions. The coefficients are determined by solving the ordinary least squares (OLS) problem

$$\min_{\boldsymbol{\beta}_i} \left\| Q_i(\mathbf{S}_i) - \sum_{j=1}^{J} \beta_{i,j}\phi_j(\mathbf{S}_i) \right\|_2^2 \tag{11}$$

The realizations of $Q_i(\mathbf{S}_i)$ for the regression are the cash flows of each path. The cash flow on path m is given by

$$Q_i^{(m)}\left(\mathbf{S}_i^{(m)}\right) = D_{i,\tau^*_{i+1,m}} h_{\tau^*_{i+1,m}}\left(\mathbf{S}^{(m)}_{\tau^*_{i+1,m}}\right),\ m = 1, ..., M. \tag{12}$$

The regression-later method is another way of estimating the continuation values. The idea is the same to the regression-now method, except that the regression is done at the next time-step. Based on the fact that $Q_i(\mathbf{S}_i) = E[D_{i,i+1}V_{i+1}(\mathbf{S}_{i+1})|\mathbf{S}_i]$, the regression-later method first estimates

$$\widehat{V}_{i+1}(\mathbf{S}_{i+1}) := \boldsymbol{\phi}^\top(\mathbf{S}_{i+1})\boldsymbol{\beta}_i = \sum_{j=1}^{J} \beta_{i,j}\phi_j(\mathbf{S}_{i+1}) \tag{13}$$

at t_{i+1} by regression, and then calculate the conditional expectation to obtain the continuation value; i.e.,

$$\widehat{Q}_i(\mathbf{S}_i) = E[D_{i,i+1}\widehat{V}_{i+1}(\mathbf{S}_{i+1})|\mathbf{S}_i] = D_{i,i+1}\sum_{j=1}^{J} \beta_{i,j}E[\phi_j(\mathbf{S}_{i+1})|\mathbf{S}_i], \tag{14}$$

where the expectation $E[\phi_j(\mathbf{S}_{i+1})|\mathbf{S}_i]$ needs to be analytically calculated. The coefficients are determined by solving the ordinary least squares (OLS) problem with L^2 norm

$$\min_{\boldsymbol{\beta}_i} \left\| V_{i+1}(\mathbf{S}_{i+1}) - \sum_{j=1}^{J} \beta_{i,j}\phi_j(\mathbf{S}_{i+1}) \right\|_2^2. \tag{15}$$

The realizations of $V_{i+1}(\mathbf{S}_{i+1})$ for the regression on path m is given by

$$V_{i+1}^{(m)}\left(\mathbf{S}_{i+1}^{(m)}\right) = h_{\tau_{i+1,m}^*}\left(\mathbf{S}_{\tau_{i+1,m}^*}^{(m)}\right),\ m = 1, ..., M. \tag{16}$$

The regression-later method (e.g., Jain and Oosterlee (2015)) usually has lower variance than the regression-now method (e.g., Longstaff and Schwartz (2001)) and it can calculate the Greeks rapidly. However, one drawback is that one needs to choose the basis functions and calculate the conditional expectations of them in closed forms. The calculation of the conditional expectations is not trivial and it is only applicable to particular stochastic processes and functional forms.

3. Kernel ridge regression

Regression based methods do not have an objective way of choosing the basis functions in the regression step, especially for high-dimensional options. If we choose the basis including all possible polynomials of the d elements of $\mathbf{S}_i$ up to the nth degree, there are

$$C_{d+n}^n = \frac{(d+n)!}{d!n!} \tag{17}$$

basis functions. E.g., for $d = 17$ and $n = 3$, the number is $C_{20}^3 > 10^3$, which is too large to handle in practice and the computation will be very time consuming. Kernel ridge regression (KRR) can be utilized to avoid the basis function selection step, so as to solve the issue of a huge number of basis functions for high-dimensional options.

Generally, consider the following linear regression model

$$y = \boldsymbol{\phi}^\top(\mathbf{x})\boldsymbol{\beta} + \epsilon,\ \mathbf{x} \in \mathbb{R}^d, \tag{18}$$

where ϵ is an independent zero-mean random noise, and the dataset

$$\boldsymbol{\Phi} = \begin{pmatrix} \boldsymbol{\phi}^\top(\mathbf{x}^{(1)}) \\ \boldsymbol{\phi}^\top(\mathbf{x}^{(2)}) \\ ... \\ \boldsymbol{\phi}^\top(\mathbf{x}^{(M)}) \end{pmatrix}_{M\times J} \quad \mathbf{y} = \begin{pmatrix} y^{(1)} \\ y^{(2)} \\ ... \\ y^{(M)} \end{pmatrix}_{M\times 1}. \tag{19}$$

The ridge regression problem on estimating $\boldsymbol{\beta}$ is

$$\min_{\boldsymbol{\beta}} (\boldsymbol{\Phi}\boldsymbol{\beta} - \mathbf{y})^\top(\boldsymbol{\Phi}\boldsymbol{\beta} - \mathbf{y}) + \lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}, \tag{20}$$

where $\lambda\boldsymbol{\beta}^\top\boldsymbol{\beta}$ is a ridge penalty term to overcome over-fitting or ill-posed problems. The corresponding OLS estimation of $\boldsymbol{\beta}$ with ridge parameter λ is

$$\widehat{\boldsymbol{\beta}} = (\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}, \tag{21}$$

and the corresponding prediction at $\mathbf{x}$ is

$$\widehat{y}(\mathbf{x}) = \boldsymbol{\phi}^\top(\mathbf{x})\widehat{\boldsymbol{\beta}} = \boldsymbol{\phi}^\top(\mathbf{x})(\boldsymbol{\Phi}^\top\boldsymbol{\Phi} + \lambda\mathbf{I})^{-1}\boldsymbol{\Phi}^\top\mathbf{y}. \tag{22}$$

The linear regression model (18) satisfies LSP (Learning subspace property, see Kung (2014) pp.11) so that the solution of $\boldsymbol{\beta}$ can be restricted to the space of span$\{[\boldsymbol{\Phi}]\}$ (the empirical space);

i.e.

$$\boldsymbol{\beta} = \boldsymbol{\Phi}^\top \boldsymbol{\alpha}, \tag{23}$$

for some $\boldsymbol{\alpha} \in R^M$. Then the optimization problem (20) can be converted to

$$\min_{\boldsymbol{\alpha}} (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \boldsymbol{\alpha} - \mathbf{y})^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top \boldsymbol{\alpha} - \mathbf{y}) + \lambda \boldsymbol{\alpha}^\top \boldsymbol{\Phi}\boldsymbol{\Phi}^\top \boldsymbol{\alpha}. \tag{24}$$

Let $\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top$, the estimation of $\boldsymbol{\alpha}$ is

$$\widehat{\boldsymbol{\alpha}} = (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y}. \tag{25}$$

As a result,

$$\widehat{\boldsymbol{\beta}} = \boldsymbol{\Phi}^\top \widehat{\boldsymbol{\alpha}} = \boldsymbol{\Phi}^\top (\mathbf{K} + \lambda \mathbf{I})^{-1} \mathbf{y} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I})^{-1} \boldsymbol{\Phi}^\top \mathbf{y}, \tag{26}$$

where the last equality uses the fact that

$$\boldsymbol{\Phi}^\top (\boldsymbol{\Phi}\boldsymbol{\Phi}^\top + \lambda \mathbf{I})^{-1} = (\boldsymbol{\Phi}^\top \boldsymbol{\Phi} + \lambda \mathbf{I})^{-1} \boldsymbol{\Phi}^\top. \tag{27}$$

It seems that KRR is not any different from OLS as the two solutions are exactly the same. However, the strength of KRR lies in the kernel trick. Notice that

$$\mathbf{K} = \boldsymbol{\Phi}\boldsymbol{\Phi}^\top, \ K_{ij} = \boldsymbol{\phi}^\top(\mathbf{x}^{(i)}) \boldsymbol{\phi}(\mathbf{x}^{(j)}), \ i, j = 1, ..., M. \tag{28}$$

$\mathbf{K}$ would be difficult to calculate if the intrinsic space (the space generated by $\boldsymbol{\phi}$) is of high dimension. Now the kernel trick will help with this. We can define a kernel function $\mathcal{K}(\mathbf{x}, \mathbf{y})$ satisfying

$$\mathcal{K}(\mathbf{x}^{(i)}, \mathbf{x}^{(j)}) = K_{ij} = \boldsymbol{\phi}^\top(\mathbf{x}^{(i)}) \boldsymbol{\phi}(\mathbf{x}^{(j)}), \tag{29}$$

which converts the inner product calculation in the high-dimensional intrinsic space (with dimension J) to the calculation in the low-dimensional original input space (with dimension d). By defining different kernel functions, the original input space can be expanded to much higher or even infinite dimensional intrinsic spaces, which is called the kernel trick. One typical kernel is the Gaussian kernel:

$$\mathcal{K}(\mathbf{x}, \mathbf{y}) = e^{-\frac{1}{C}(\mathbf{x}-\mathbf{y})^\top (\mathbf{x}-\mathbf{y})}, \ C > 0, \tag{30}$$

resulting in an infinite-dimensional intrinsic space. Theoretically, the Gaussian kernel can cover any continuous basis functions, because $\boldsymbol{\phi}(\mathbf{x})$ for the Gaussian kernel contains the dampened polynomial

$$e^{-\frac{1}{C}\mathbf{x}^\top \mathbf{x}} \prod_{i=1}^{d} \frac{\mathbf{x}_i^{n_i}}{\sqrt{n_i!}} \tag{31}$$

for each combination of nonnegative degrees $n_1, ..., n_d$ (Exterkate (2013)). It is expected that KRR with the Gaussian kernel can give better prediction results than using limited basis functions. Under KRR with kernel tricks, the predicted function value at $\mathbf{x}$ becomes

$$\widehat{\mathbf{y}}(\mathbf{x}) = \boldsymbol{\phi}^\top(\mathbf{x}) \widehat{\boldsymbol{\beta}} = \boldsymbol{\phi}^\top(\mathbf{x}) \boldsymbol{\Phi}^\top \widehat{\boldsymbol{\alpha}} = \sum_{m=1}^{M} \widehat{\boldsymbol{\alpha}}_m \mathcal{K}(\mathbf{x}^{(m)}, \mathbf{x}). \tag{32}$$

In summary, the discussion above is an outline of KRR, which is the proposed approach to avoid choosing basis functions for regression-based methods. Please note that by means of KRR, the problem of choosing basis functions is actually transferred into one of choosing kernel functions, which is much more tractable.

4. Application of KRR to regression-based methods

4.1. *Regression-now*

We first consider applying KRR to regression-now methods. It is actually straightforward as we only need to change the regression to KRR and estimate the continuation value at $\mathbf{S}_i$ as

$$\widehat{Q}_i(\mathbf{S}_i) = D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{i,m} \mathcal{K}(\mathbf{S}_i^{(m)}, \mathbf{S}_i). \tag{33}$$

The problem worthy of consideration is the implementation efficiency. As the basic method we are using for pricing is Monte Carlo, thousands of paths are needed in the simulation. This leads to regressions with large-scale samples, which finally results in a large-scale matrix $\mathbf{K}$ (defined in (28)) in KRR. The calculation of matrix inversion for estimating $\widehat{\boldsymbol{\alpha}}$ would be very slow. Furthermore, the regression needs to be done at every time-step. KRR cannot be naively applied to the regression-based method.

The main idea of the solution is to partition the global regression into small-scale local regressions as proposed in Zhang *et al.* (2013). Suppose the whole dataset is partitioned into P sub-datasets. Then the time complexity of the regression can be reduced from $O(M^3)$ to $O(M^3/P^2)$. In light of the specific problem of pricing, it is better to use bundling (Jain and Oosterlee (2015)) instead of the random partition in Zhang *et al.* (2013). The basic idea of bundling is to group together similar grid points (simulated underlying asset prices) at the same time-step to get better regression results. There are many approaches for bundling, such as k-means clustering and sorting the data by bundling references. We will see that KRR can provide a better fit and become more efficient due to bundling.

Specifically, suppose the current step is to estimate the continuation values by regression at t_i. In order to do local regression, we first use the payoff function as the bundling reference and partition the grid points at t_{i-1} into P non-overlapping bundles with $\widetilde{M} = \lfloor M/P \rfloor$ paths in each bundle. The regression is then performed at t_i for the paths belong to the same bundle. Local regression at bundle level has been proved more accurate than global regression (Jain and Oosterlee (2015)), due to the similarity among the samples. In other words, bundling improves not only the efficiency of KRR, but also the accuracy provided P is not too large.

4.2. *Regression-later*

For regression-later, we use the Gaussian kernel (30) as the kernel function and perform KRR on $\mathbf{X}_t = \ln \mathbf{S}_t$ instead of $\mathbf{S}_t$. Within the framework of the regression-later method with KRR, the estimation of the continuation value at $\mathbf{X}_i$ can be expressed as

$$\begin{aligned} \widehat{Q}_i(\mathbf{S}_i) &= D_{i,i+1} E\left[\sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} \mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1}) \Big| \mathbf{X}_i \right] \\ &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} E[\mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1}) | \mathbf{X}_i]. \end{aligned} \tag{34}$$

As a result, the problem of calculating the conditional expectations of the basis function in (14) is converted into calculating the conditional expectations of the kernel function. The benefit is that we not only avoid the choice of basis functions, but also may simplify the expectation calculation, as the there is only one expectation to calculate. For example, the expectation of the payoff function of max or min options has no analytic solution, so that approximations are needed if the payoff function is selected as the basis function (E.g., Jain and Oosterlee (2015) use Clark's algorithm to do the approximation). In contrast, no such an issue will arise in this case when using KRR. In practice, we also perform local regression by bundling for the regression-later method.

4.2.1. **Regression-later under geometric Brownian motion.** We first consider the most common situation in which the underlying assets follow the multi-dimensional geometric Brownian motion (GBM)

$$\frac{dS_{t,\nu}}{S_{t,\nu}} = (r - q_\nu)dt + \sigma_\nu W_{t,\nu},\ \nu = 1, ..., d, \tag{35}$$

where r is the risk-free interest rate, q_ν are the dividend rates and $W_{t,\nu}$ are standard Brownian motions with instantaneous correlation coefficient ρ_{ij} between $W_{t,i}$ and $W_{t,j}$, $i, j = 1, ..., d$. Define a $d \times d$ matrix $\mathbf{\Sigma}^S$ with $\Sigma^S_{ij} = \sigma_i \sigma_j \rho_{ij}$ and let $\mathbf{A}\mathbf{A}^{'} = \mathbf{\Sigma}^S$. By denoting $\mathbf{X}_t = \ln \mathbf{S}_t$, we can simulate the underlying assets as

$$X_{i+1,\nu} = X_{i,\nu} + (r - q_\nu - \frac{1}{2}\sigma_\nu^2)h + \sqrt{h}\sum_{j=1}^{d} A_{\nu j} Z_{i,j},\ i = 0, ..., N-1, \tag{36}$$

where $Z_{i,j}$ are independent standard normal random variables, and $h = T/N$ is the step size.

Lemma 1 *Suppose* $\mathbf{X} \sim N(\boldsymbol{\mu}, \mathbf{\Sigma})$. *Then*

$$E\left[e^{-\frac{1}{C}\mathbf{X}^\top \mathbf{X}}\right] = \frac{1}{|\frac{2}{C}\mathbf{\Sigma} + \mathbf{I}|^{1/2}} e^{-\frac{1}{C}\boldsymbol{\mu}^\top \left(\frac{2}{C}\mathbf{\Sigma} + \mathbf{I}\right)^{-1}\boldsymbol{\mu}}. \tag{37}$$

Proof. See Appendix A. □

Based on Lemma 1, we have the following result.

Proposition 1 *Suppose the underlying assets follow the GBM model (35), and the Gaussian kernel is used in KRR. The conditional expectation in (34) is given by*

$$\begin{aligned} \widehat{Q}_i(\mathbf{X}_i) &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} E\left[\mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1}) | \mathbf{X}_i\right] \\ &= \frac{D_{i,i+1}}{|\frac{2h}{C}\mathbf{\Sigma}^S + \mathbf{I}|^{1/2}} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} e^{-\frac{1}{C}\left(\boldsymbol{\mu}_{i+1} - \mathbf{X}_{i+1}^{(m)}\right)^\top \left(\frac{2h}{C}\mathbf{\Sigma}^S + \mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_{i+1} - \mathbf{X}_{i+1}^{(m)}\right)}, \end{aligned} \tag{38}$$

where $\boldsymbol{\mu}_{i+1} = (\mu_{i+1,1}, ..., \mu_{i+1,d})^\top$, *and*

$$\mu_{i+1,\nu} = X_{i,\nu} + (r - q_\nu - \frac{1}{2}\sigma_\nu^2)h,\ \nu = 1, ..., d. \tag{39}$$

Proof. See Appendix B. □

Corollary 1 *Suppose the conditions in Proposition 1 are satisfied and the underlying assets are independent from each other. Then the conditional expectation in (34) is given by*

$$\widehat{Q}_i(\mathbf{X}_i) = \frac{D_{i,i+1}C^{d/2}}{\sqrt{\prod_{\nu=1}^{d}(2\sigma_\nu^2 h + C)}} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} \exp\left(-\sum_{\nu=1}^{d} \frac{\left(\mu_{i+1,\nu} - X_{i+1,\nu}^{(m)}\right)^2}{2\sigma_\nu^2 h + C} \right). \tag{40}$$

Proof. When the underlying assets are independent from each other, $\mathbf{\Sigma}^S$ is a diagonal matrix and the result can be directly verified based on Proposition 1. □

4.2.2. **Regression-later under Merton jump diffusion.** We then consider the multi-dimensional Merton jump diffusion (MJD) process driven by a common Poisson process. The model is

$$\begin{cases} \dfrac{dS_{t,\nu}}{S_{t,\nu}} = (r - q_\nu - \lambda^J \kappa_\nu)dt + \sigma_\nu W_{t,\nu} + d\Gamma_t, \ \nu = 1, ..., d \\ \Gamma_t = \displaystyle\sum_{n=1}^{N(t)} \left(e^{Z_{\nu,n}^J} - 1 \right) \end{cases}, \tag{41}$$

where the diffusion part is the same as for the GBM process (35), Γ_t is a compound Poisson process with jump intensity λ^J, $\mathbf{Z}_n^J = (Z_{1,n}^J, ..., Z_{d,n}^J)^\top$. are the jump sizes following multi-dimensional normal distribution $N(\boldsymbol{\mu}^J, \mathbf{\Sigma}^J)$ with $\boldsymbol{\mu}^J = (\mu_1^J, ..., \mu_d^J)^\top$ and $\Sigma_{ij}^J = \sigma_i^J \sigma_j^J \rho_{ij}^J$, $\kappa_\nu = E[e^{Z_{\nu,n}^J} - 1] = \exp(\mu_\nu^J + (\sigma_\nu^J)^2/2) - 1$ so as to make $e^{-rt}S_{t,\nu}$ a martingale under the risk neutral measure. The processes W_t, N_t and $\mathbf{Z}_n^J$ are assumed to be independent of each other, and $\mathbf{Z}_n^J$ are i.i.d. In the model (41), the jump times are the same for different components of $\mathbf{S}_t$, but the jump sizes are different.

The underlying asset $S_{t,\nu}$ has the analytic solution

$$S_{t,\nu} = S_{0,\nu} \exp\left((r - q_\nu - \lambda^J \kappa_\nu - \frac{1}{2}\sigma_\nu^2)t + \sigma_\nu W_{t,\nu} \right) \exp\left(\sum_{n=1}^{N(t)} Z_{\nu,n}^J \right), \tag{42}$$

where $N(t)$ is the number of jumps in $[0, t]$. Correspondingly,

$$\begin{aligned} X_{t,\nu} &= \ln S_{t,\nu} \\ &= \ln S_{0,\nu} + (r - q_\nu - \lambda^J \kappa_\nu - \frac{1}{2}\sigma_\nu^2)t + \sigma_\nu W_{t,\nu} + \sum_{n=1}^{N(t)} Z_{\nu,n}^J. \end{aligned} \tag{43}$$

Denote ΔN_i as the number of jumps in $(t_i, t_{i+1}]$, then conditional on $\mathbf{X}_i$ and $\Delta N_i = k$, $\mathbf{X}_{i+1}$ follows the multi-dimensional normal distribution

$$\mathbf{X}_{i+1} \sim N\left(\boldsymbol{\mu}_{i+1}^{JD}, \mathbf{\Sigma}^{JD} \right), \tag{44}$$

where

$$\mu_{i+1,\nu}^{JD} = X_{i,\nu} + (r - q_\nu - \lambda^J \kappa_\nu - \frac{1}{2}\sigma_\nu^2)h + k\mu_\nu^J, \ \nu = 1, ..., d, \tag{45}$$

and $\mathbf{\Sigma}^{JD} = h\mathbf{\Sigma}^S + k\mathbf{\Sigma}^J$.

Proposition 2 *Suppose the underlying assets follow the MJD model (41), and the Gaussian kernel is used in KRR. The conditional expectation in (34) is given by*

$$\begin{aligned}\widehat{Q}_i(\mathbf{X}_i) &= D_{i,i+1}\sum_{m=1}^{M}\widehat{\alpha}_{i+1,m}E\left[\mathcal{K}(\mathbf{X}_{i+1}^{(m)},\mathbf{X}_{i+1})|\mathbf{X}_i\right] \\ &= D_{i,i+1}\sum_{k=0}^{\infty}\frac{e^{-\lambda^J h}(\lambda^J h)^k}{k!|\frac{2}{C}\mathbf{\Sigma}^{JD}+\mathbf{I}|^{\frac{1}{2}}}\sum_{m=1}^{M}\widehat{\alpha}_{i+1,m}e^{-\frac{1}{C}\left(\boldsymbol{\mu}_{i+1}^{JD}-\mathbf{X}_{i+1}^{(m)}\right)^\top\left(\frac{2}{C}\mathbf{\Sigma}^{JD}+\mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_{i+1}^{JD}-\mathbf{X}_{i+1}^{(m)}\right)}.\end{aligned} \tag{46}$$

Proof. See Appendix C. □

In practice, we need to do truncation when calculating the infinite sum in (46).

4.3. *Regression-later for computing the Greeks*

As pointed out in Jain and Oosterlee (2015), regression-later with bundling can be conveniently used to calculate the Delta and Gamma of American options. Specifically,

$$\begin{aligned}\Delta^\nu &= \frac{\partial V_0(\mathbf{S}_0)}{\partial S_0^\nu} \approx \frac{\partial \widehat{Q}_0(\mathbf{S}_0)}{\partial S_0^\nu} = D_{0,1}\sum_{m=1}^{M}\widehat{\alpha}_{1,m}\frac{\partial E\left[\mathcal{K}(\mathbf{X}_1^{(m)},\mathbf{X}_1)|\mathbf{X}_0\right]}{\partial X_0^\nu}\frac{1}{S_0^\nu}, \\ \Gamma^\nu &= \frac{\partial^2 V_0(\mathbf{S}_0)}{(\partial S_0^\nu)^2} \approx \frac{\partial^2 \widehat{Q}_0(\mathbf{S}_0)}{(\partial S_0^\nu)^2} = \frac{\partial \Delta^\nu}{\partial X_0^\nu}\frac{1}{S_0^\nu},\ \nu = 1,...,d.\end{aligned} \tag{47}$$

Furthermore, based on the explicit expression for $E\left[\mathcal{K}(\mathbf{X}_1^{(m)},\mathbf{X}_1)|\mathbf{X}_0\right]$, we have the following expressions for the approximations.

Proposition 3 *Suppose the underlying assets follow the GBM model (35), and the Gaussian kernel is used in KRR. The Delta of the option is given by*

$$\Delta^\nu \approx \frac{\partial \widehat{Q}_0(\mathbf{S}_0)}{\partial S_0^\nu} = \frac{D_{0,1}}{S_0^\nu|\frac{2h}{C}\mathbf{\Sigma}^S+\mathbf{I}|^{1/2}}\sum_{m=1}^{M}\widehat{\alpha}_{1,m}\gamma_{m,\nu}e^{-\frac{1}{C}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)^\top\left(\frac{2h}{C}\mathbf{\Sigma}^S+\mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)}, \tag{48}$$

and the Gamma of the option is given by

$$\begin{aligned}\Gamma^\nu \approx \frac{\partial \Delta^\nu}{\partial X_0^\nu}\frac{1}{S_0^\nu} =& \frac{D_{0,1}}{(S_0^\nu)^2|\frac{2h}{C}\mathbf{\Sigma}^S+\mathbf{I}|^{1/2}} \\ &\sum_{m=1}^{M}\widehat{\alpha}_{1,m}\left(\Sigma_{\nu\nu}^\gamma+\gamma_{m,\nu}^2-\gamma_{m,\nu}\right)e^{-\frac{1}{C}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)^\top\left(\frac{2h}{C}\mathbf{\Sigma}^S+\mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)},\end{aligned} \tag{49}$$

where $\boldsymbol{\gamma}_m = \mathbf{\Sigma}^\gamma\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)$ *and* $\mathbf{\Sigma}^\gamma = -\frac{2}{C}\left(\frac{2h}{C}\mathbf{\Sigma}^S+\mathbf{I}\right)^{-1}$.

Proof. See Appendix D. □

The computation of the Greeks discussed above is based on the KRR regression results at t_1, so that it is no more difficult than pricing, and little extra effort is needed. The constraint is that

it is only applicable to the sensitivity with respect to the underlying assets. With regard to the implementation, there is only one bundle at time t_0, but we can still do local regression by randomly partitioning the grid points at time t_1. Therefore, each bundle at time t_1 can give one estimation of the Greeks, and we then take the average.

4.4. *Hyperparameters selection*

There are two key parameters in KRR, namely, the ridge parameter λ and the Gaussian kernel parameter C. They cannot be determined from Equation (25) and must be determined from the data. We refer to such parameters as hyperparameters. The optimal hyperparameters for each regression can be selected by cross-validation with grid searching. However, the process is time consuming. What's more, as there are regressions for each bundle at each time-step in our method, we need to keep it simple and avoid selecting the hyperparameters for each time-step.

The strategy we use in this paper is to perform calibrations at the price level instead of the regression level at each time-step. In any case, the final purpose is pricing rather than regression. One of the advantages of this strategy is that once a suitable set of parameters are found, the pricing results may be not so sensitive to the change of the parameters.

Specifically, the way we determine the parameters is to always fix the ridge parameter λ to 1 just as Zhang *et al.* (2013) suggest (in Corollary 5), and then select a suitable C for all the time-steps for the same option. As discussed in Exterkate (2013), parameter C controls the smoothness of the prediction, and its optimal value depends on the dimension of the input space. We set $C = C_0$ or $C = C_1 d$, where $C_0 > 0$, $C_1 > 0$ and d is the dimension of S_t, so that there is only one parameter to be estimated. For a set of similar options with only different dimensions, C_0 is selected for all of them by a simple cross-validation strategy:

Step 1 Set a tolerance of mean squared error for the pricing, and a maximum number of iterations.
Step 2 Calibrate an option of low dimension (e.g., $d = 5$) to its true price (or a price given by LSM in case the true price is unknown) with a suitable C_0.
Step 3 Validate on an option of high dimension (e.g., $d = 100$) with the selected C_0. If the pricing result is beyond the error tolerance, go back to Step 2.

If C_0 cannot be found by the cross-validation strategy above, we turn to estimating C_1 with a similar procedure. If C_1 cannot be found either, we estimate C_0 individually for each option.

Numerical test results indicate that this strategy works well, and the pricing results are actually not sensitive to the value of C. Therefore, C is not necessarily the optimal value and the calibration process can even be done manually within several trials. Another hyperparameter is the number of bundles P. The numerical test results show that the optimal value of it is different for different pricing parameters. The quantitative rule to determine the optimal P requires further investigation. In this paper, we simply set it to a fixed value for all cases.

4.5. *Algorithm*

We summarize our method into a complete algorithm for pricing and calculating Greeks for high-dimensional American options by regression-based methods.

Algorithm 1 *Kernel ridge regression for pricing and calculating Greeks for high-dimensional American options.*

Step 1 Preprocessing. Determine the hyperparameters λ, C and P through the strategy described in Section 4.4.
Step 2 Simulate M paths with N time-steps for the underlying assets.
Step 3 Calculate the options values at t_N to obtain the initial cash flows.
Step 4 For each $i = N - 1, ..., 1$

a) Bundling. Partition the grid points at t_{i-1} (regression-now) or t_i (regression-later) into P bundles.
b) Estimating continuation values. Within each bundle, perform KRR at t_i (regression-now) or t_{i+1} (regression-later) to estimate $\widehat{\boldsymbol{\alpha}}_i$ and get $\widehat{Q}_i$ by (33) or (34).
c) Renewing cash flows. Compare the continuation values with the exercise values and renew the cash flows.

Step 5 Additional KRR for regression-later at t_0. Randomly bundle the grid points and perform KRR at t_1 to estimate $\widehat{\boldsymbol{\alpha}}_0$ and get $\widehat{Q}_0$ by (34).

Step 6 Computing the Greeks (optional). Calculate the Greeks by Proposition (3) based on $\widehat{\boldsymbol{\alpha}}_1$ generated in Step 5.

Step 7 Average the discounted cash flows at each path to get the option price.

5. Numerical test

In this section, three main aspects of our method will be tested based on various American options under GBM and MJD processes. The first is the pricing accuracy and efficiency for a range of options with different dimensions. The second is the error and computation time for the same options with a different number of bundles in KRR. The last is the robustness of our method when some of the parameters vary. For the first aspect, we will compare the pricing results with LSM. The basis functions of LSM are all the polynomials up to the second degree. We also set another LSM with the payoff function as an additional basis function. The local KRR is performed with default number of bundles $P = 100$ on the paths with positive immediate exercise values.

In order to show the strength of KRR, the options are set with very high dimensions, up to 100 underlying assets. As there are no reliable benchmark prices (either from existing literature or other pricing methods) for such high-dimensional options, we resort to Premia [1], a third-party software designed for option pricing, hedging and financial model calibration, and take its results as the benchmark. All the tests were performed by Matlab 2017b on a system with an Intel(R) Core(TM) i7-7500U 2.7 GHZ CPU and 8 GB of RAM in a Windows 10 environment.

5.1. *Max call under GBM*

We first test the proposed approach for a multi-dimensional max call American option with payoff function

$$h(\mathbf{S}_t) = (\max(S_{t,1}, ..., S_{t,d}) - K)^+, \tag{50}$$

where $\mathbf{S}_t$ follows the multi-dimensional GBM process defined by (35), and K is the strike price. The parameters are set as

$$\begin{aligned} &S_{0,\nu} = K = 100,\ \sigma_\nu = 0.2,\ r = 5\%,\ q_\nu = 10\%,\ \rho_{ij} = \rho = 0,\ T = 3, \\ &d = \{5, 10, 15, 20, 30, 40, 60, 80, 100\},\ P = 100,\ N = 3,\ M = 10,000, \end{aligned} \tag{51}$$

which are similar to the cases studied in Broadie *et al.* (2004). The same fixed λ and kernel parameter C are used for pricing all the options.

5.1.1. **Pricing accuracy and efficiency.** Table 1 and Figure 1 report the pricing results of four algorithms. Column Premia contains the benchmark prices calculated by Premia. They are

[1] https://www.rocq.inria.fr/mathfi/Premia/index.html

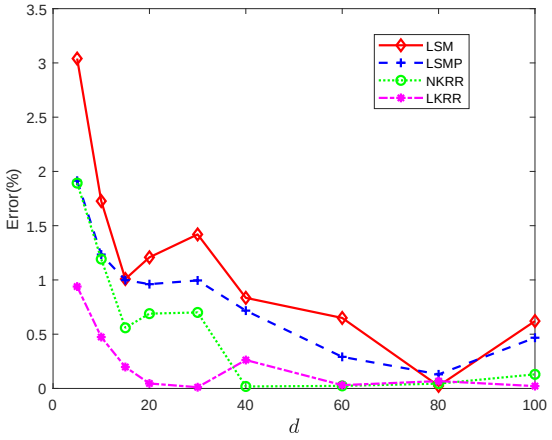
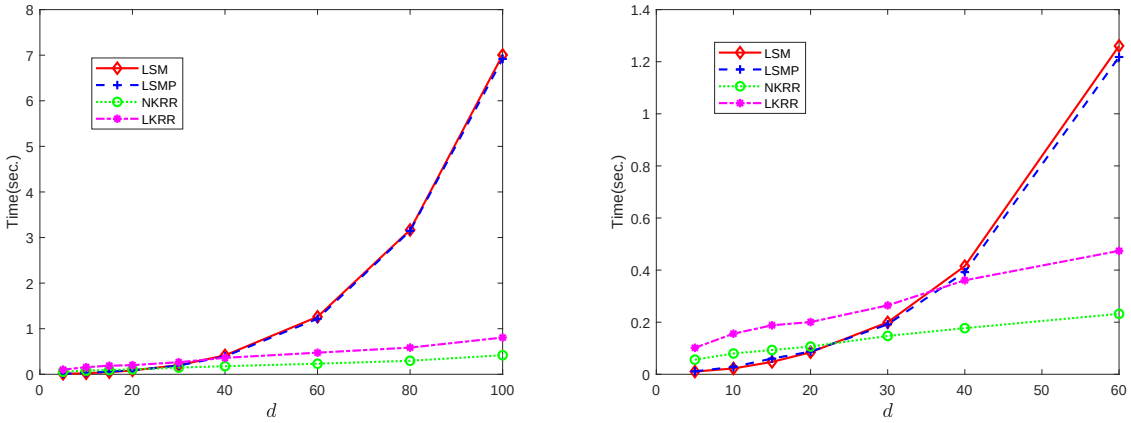


Figure 1. Pricing errors under GBM for a range of dimensions.

point estimations based on the primal-dual method in Andersen and Broadie (2004). However, Premia does not provide the confidence interval for the benchmark prices. We construct the 95% confidence interval for them by estimating the standard deviations by 10 independent runs with the same input data. Please see more details in Table 2. The columns in Table 1 and lines in the figure labelled as LSM, LSMP, NKRR and LKRR correspond to, respectively, LSM without payoff basis function, LSM with payoff basis function, KRR with regression-now, and KRR with regression-later. The values in the parentheses are the standard deviations based on 10 independent batches. All errors are relative to Premia's prices. It can be seen that generally NKRR/LKRR outperform LSM/LSMP, LKRR is the most accurate method, and LSMP is a little better than LSM.

Figure 2 shows the computation time of these algorithms. LSM/LSMP usually run fast in low-dimensional cases, which is also indicated in the figure. When $d \leq 20$, NKRR and LKRR are relatively slower. However, the computation time for LSM/LSMP seem to increase exponentially with dimension, while for NKRR/LKRR it is approximately linear. NKRR/LKRR start to exhibit the speed advantage when $d > 40$.



(a) $5 \leq d \leq 100$ (b) $5 \leq d \leq 60$ (zoomed in)

Figure 2. Computation time under GBM for a range of dimensions.

Table 1. Pricing results under GBM for a range of dimensions.

d	Premia	LSM	Error(%)	LSMP	Error(%)	NKRR	Error(%)	LKRR	Error(%)
5	25.306	24.536(0.230)	3.040	24.821(0.187)	1.914	24.827(0.287)	1.893	25.543(0.149)	**0.939**
10	37.698	37.047(0.259)	1.727	37.232(0.264)	1.234	37.248(0.255)	1.192	37.876(0.224)	**0.474**
15	45.569	45.110(0.211)	1.008	45.114(0.178)	0.999	45.315(0.180)	0.559	45.659(0.285)	**0.198**
20	51.443	50.821(0.264)	1.208	50.949(0.242)	0.961	51.088(0.276)	0.689	51.467(0.220)	**0.046**
30	59.775	58.927(0.263)	1.420	59.180(0.161)	0.995	59.357(0.257)	0.700	59.769(0.277)	**0.011**
40	65.525	64.978(0.300)	0.835	65.055(0.290)	0.717	65.514(0.236)	**0.018**	65.697(0.304)	0.262
60	73.900	73.420(0.149)	0.650	73.685(0.252)	0.291	73.918(0.297)	**0.024**	73.878(0.268)	0.031
80	79.908	79.926(0.174)	0.023	80.011(0.163)	0.130	79.943(0.245)	**0.044**	79.962(0.193)	0.068
100	84.501	85.025(0.288)	0.621	84.895(0.311)	0.467	84.610(0.264)	0.130	84.482(0.153)	**0.022**

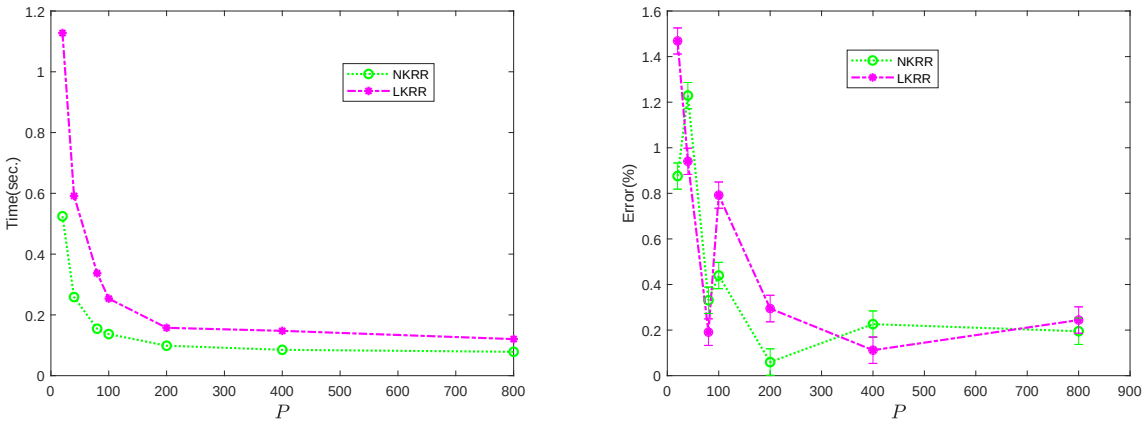
NKRR: $\lambda = 1, C = 10^5$; LKRR: $\lambda = 1, C = 30$; Other parameters see (51).

Table 2. 95% confidence interval of the benchmark prices from Premia.

d	5	10	15	20	30	40	60	80	100
Premia	25.306	37.698	45.569	51.443	59.775	65.525	73.900	79.908	84.501
STD	0.073	0.079	0.087	0.110	0.056	0.091	0.129	0.090	0.097
LCL	25.261	37.649	45.515	51.375	59.740	65.468	73.820	79.852	84.441
UCL	25.351	37.747	45.623	51.511	59.810	65.582	73.980	79.964	84.561

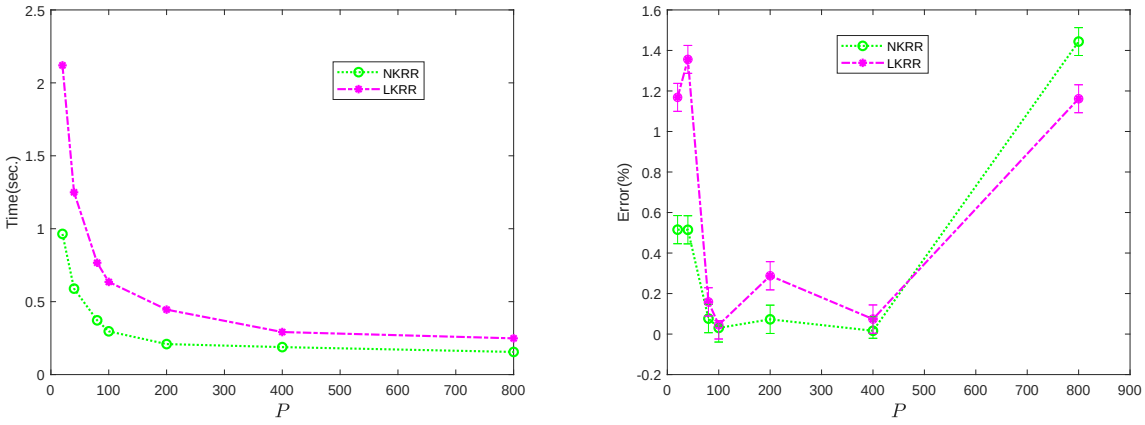
Notes. The benchmark prices are point estimations calculated by the primal-dual method in Andersen and Broadie (2004) from Premia. The primal early-exercise policy is obtained by LSM with 50000 simulation paths and Hermite polynomials of degree $0-3$ as the basis functions. The dual prices are calculated with 500 outer simulation paths and 100 nested simulation paths. STD is the standard deviation of 10 runs, $\text{LCL} = \text{Premia} - 1.96 * \text{STD}/\sqrt{10}$ is the 95% lower confidence limit and $\text{UCL} = \text{Premia} + 1.96 * \text{STD}/\sqrt{10}$ is the 95% upper confidence limit.

5.1.2. **The influence of the number of bundles.** As local KRR runs faster with a larger number of bundles P under the same number of paths M, we can accelerate NKRR/LKRR by increasing P. The test results in Jain and Oosterlee (2015) show that the pricing error decreases when the number of bundles increases. (However, the computation time will increase in their test.) Combining these two, we can expect that NKRR/LKRR may run faster and with higher accuracy as the number of bundles increases. Figures 3 and 4 display the corresponding test results. The error bars in Figures 3(b) and 4(b) are calculated according to the confidence intervals of the benchmark prices in Table 2 (similarly for the error bars in Figures 5 and 6 below). Figure 3 confirms our guess and the error and running time both decrease as the number of bundles increases for $d = 30$. However, Figure 4 gives the negative result that the error decreases first and increases after $P > 200$. The reason is that with P increasing, there are fewer regression points in each bundle, which may lead to higher estimation errors.



(a) Computation time (b) Pricing error

Figure 3. Pricing results under GBM for a range of values of P with $d = 30$.



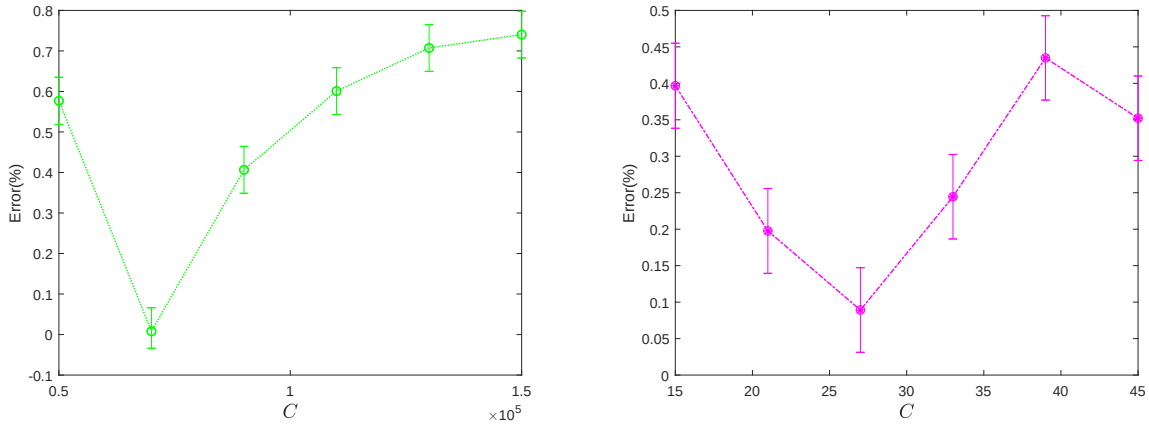
(a) Computation time (b) Pricing error

Figure 4. Pricing results under GBM for a range of values of P with $d = 100$.

The results indicate that an optimal P exists, but it is different for options with different parameters. It would be interesting to develop an algorithm that can quickly find the optimal P. Incidentally, we also did local regression for LSM and LSMP, but we found it could not improve the pricing accuracy, i.e., $P = 1$ is the best choice for LSM/LSMP.

5.1.3. **The robustness test.** We use the same λ and C for all the pricing in the same KRR algorithm. The value of C is selected through calibration and then set fixed to price all the options. There are two potential concerns. One is whether the price results are sensitive to the value of C and λ, and the other is whether a same C is able to price options with different parameters. The aim of the test in this section is to dispel these doubts.

For the first concern, from the theoretical perspective we know that when C changes, the coefficients $\widehat{\boldsymbol{\alpha}}$ (equation (25)) will change accordingly to make the result accurate. We also perform a numerical test to further confirm this. Figure 5 presents the results for both NKRR and LKRR for $d = 30$. Note that we originally set $C = 10^5$ and $C = 30$, respectively, for NKRR and LKRR in all the pricing, and now let C vary from $0.5C$ to $1.5C$. The errors are all below 0.8% for NKRR, and below 0.5% for LKRR. We can also see from the figures that our selection of C is not optimal (but it is sufficient to give accurate results) and the pricing results of our method in Table 1 can actually be improved by selecting a more suitable C.



(a) NKRR. (b) LKRR.

Figure 5. Pricing results under GBM for a range of values of C with $d = 30$.

We performed the same analysis for λ. The rule we originally use is to simply set $\lambda = 1$ for both NKRR and LKRR. Now we let λ vary from 0.5 to 1.5 and report the results in Figure 6. From the figure we can see that $\lambda = 1$ is nearly optimal for the interval $[0.5, 1.5]$. The test results confirm that the rule for setting λ is reasonable for the test cases in this section.

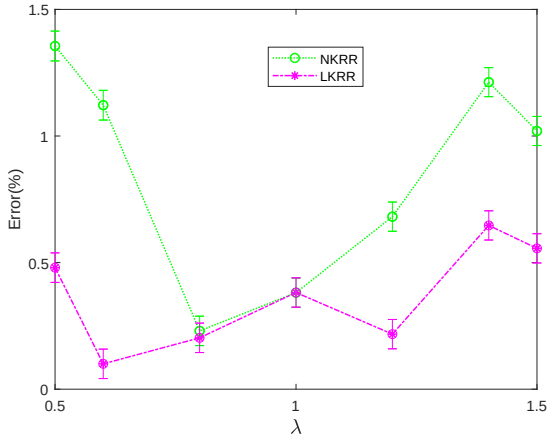
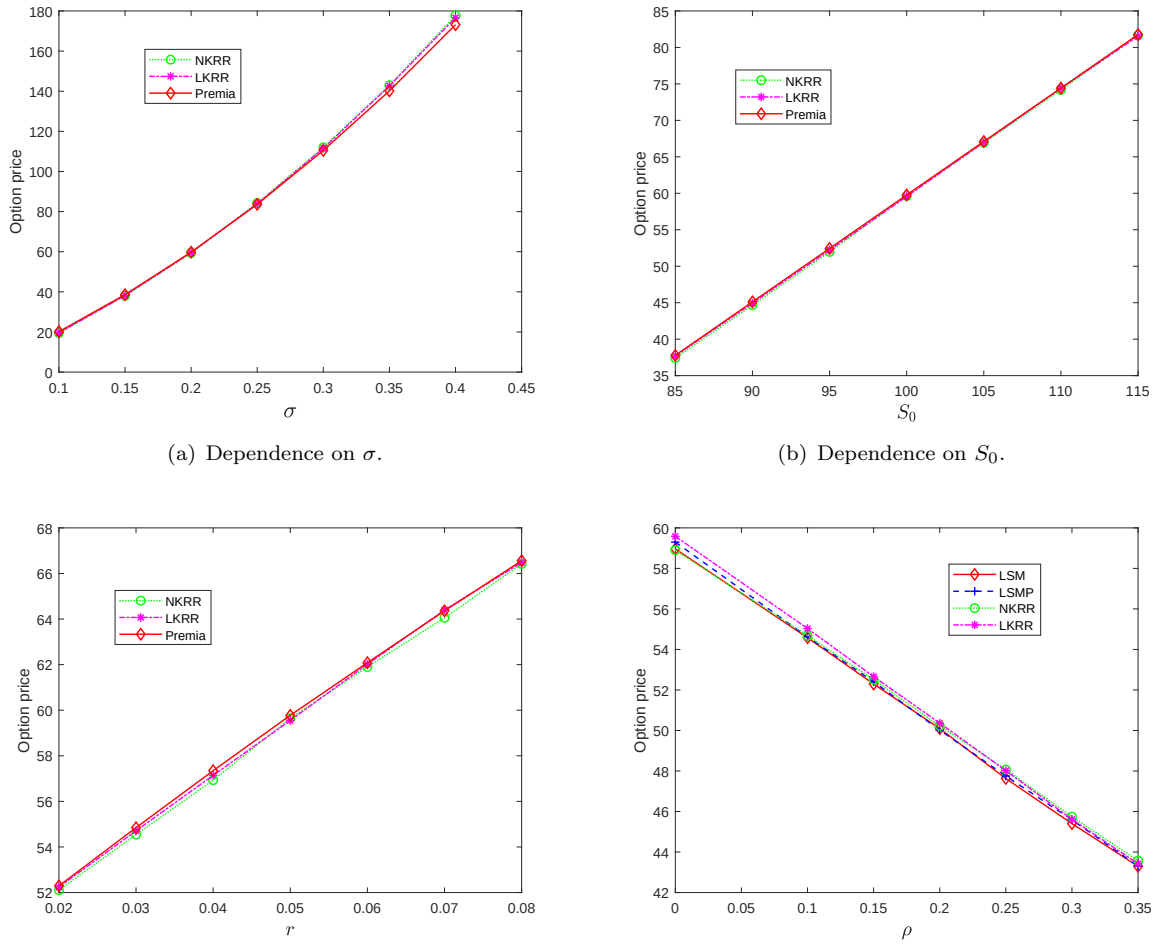


Figure 6. Pricing results under GBM for a range of values of λ with $d = 30$.

For the other concern, Figure 7 reports the pricing results using the same C when one of the parameters σ, S_0, r or ρ changes. For the test when ρ changes, we use the prices of LSM and LSMP because the prices of Premia are unavailable for $\rho > 0$. We can see that both NKRR and LKRR can give accurate results in all the cases. The results show that our method in determining the hyper parameters is reasonable and robust.



(a) Dependence on σ.

(b) Dependence on S_0.

(c) Dependence on r.

(d) Dependence on ρ.

Figure 7. Pricing results under GBM for various parameters with $d = 30$.

5.2. *Geometric basket put under MJD*

In this section, we test the proposed approach on high-dimensional geometric put American options with payoff function

$$h(\mathbf{S}_t) = \left(K - \prod_{\nu=1}^{d} S_{t,\nu}^{\frac{1}{d}} \right)^+ . \tag{52}$$

The prices $\mathbf{S}_t$ follows the multi-dimensional MJD process defined by (41).

5.2.1. **Test case design.** As there is no reliable price reference for high-dimensional geometric American options under MJD (in particular, Premia 16, which we use, does not support multi-

dimensional MJD), we need to design the test cases carefully. Notice that the multi-dimensional problem can be converted to the one-dimensional problem

$$\widetilde{S}_t = \widetilde{S}_0 \exp\left((r - \widetilde{q} - \widetilde{\lambda}^J\widetilde{\kappa} - \frac{1}{2}\widetilde{\sigma}^2)t + \widetilde{\sigma}W_t\right)\exp\left(\sum_{n=1}^{N(t)}\widetilde{Z}_n^J\right), \tag{53}$$

where

$$\begin{aligned}
&\widetilde{S}_t = \prod_{\nu=1}^{d} S_{t,\nu}^{\frac{1}{d}},\ \widetilde{\kappa} = \exp\left(\widetilde{\mu}^J + \frac{1}{2}(\widetilde{\sigma}^J)^2\right) - 1,\ \widetilde{\lambda}^J = \lambda^J,\\
&\widetilde{\mu}^J = \frac{1}{d}\sum\nolimits_{\nu}\mu_\nu^J,\ \widetilde{\sigma}^J = \frac{1}{d}\left(\sum\nolimits_{i,j}\sigma_i^J\sigma_i^J\rho_{ij}^J\right)^{\frac{1}{2}},\\
&\widetilde{\sigma} = \frac{1}{d}\left(\sum\nolimits_{i,j}\sigma_i\sigma_i\rho_{ij}\right)^{\frac{1}{2}},\ \widetilde{q} = \frac{1}{d}\sum\nolimits_{\nu}(q_\nu + \frac{1}{2}\sigma_\nu^2 + \lambda^J\kappa_\nu) - \frac{1}{2}\widetilde{\sigma}^2 - \lambda^J\widetilde{\kappa}.
\end{aligned} \tag{54}$$

The parameters for the one-dimensional problem are set as

$$\begin{aligned}
&\widetilde{S}_0 = K = 40,\ \widetilde{\sigma} = \widetilde{\sigma}^J = \sqrt{0.05},\ r = 8\%,\ \widetilde{q} = 0,\\
&\widetilde{\mu}^J = -0.025,\ \widetilde{\lambda}^J = 5,\ T = 1,\ N = 10,
\end{aligned} \tag{55}$$

which are the same as in the cases studied in Broadie and Yamamoto (2003). The true price is 6.995, and based on this we can solve an inverse problem and design high-dimensional geometric options that can be converted exactly into this one-dimensional option. Below is one of the solutions:

$$\begin{aligned}
&S_0 = K = \widetilde{S}_0,\ \sigma_\nu = \sigma_\nu^J = a\widetilde{\sigma},\ r = 8\%,\ \mu_\nu^J = \widetilde{\mu}^J,\ \lambda^J = \widetilde{\lambda}^J,\\
&q_\nu = \frac{1}{2}(\widetilde{\sigma}^2 - \sigma_\nu^2) - \lambda^J\kappa_\nu,\ \rho_{ij} = \rho_{ij}^J = \frac{d/a^2 - 1}{d-1},\ a > 1,
\end{aligned} \tag{56}$$

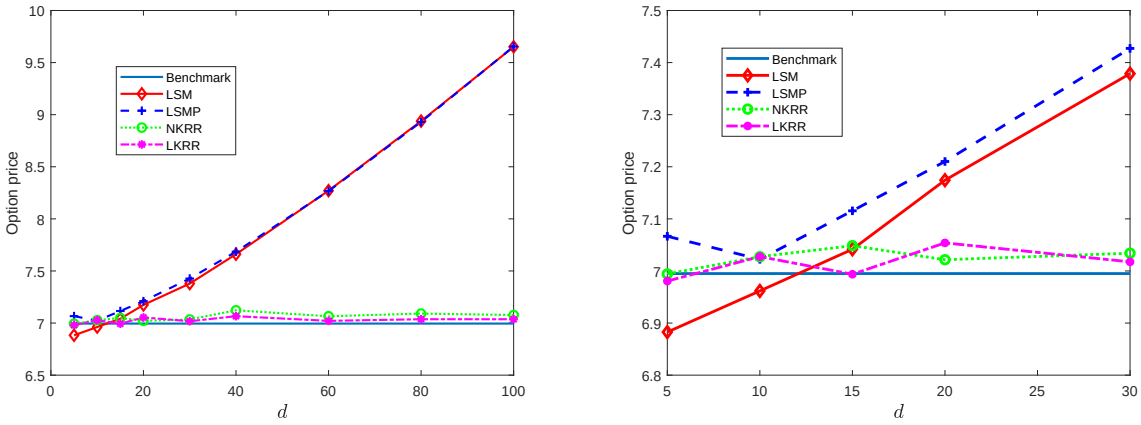
where we set $a = 1.5$. It can easily be verified that the parameter settings in (56) can be converted into (55) according to the relationship described in (54) (notice that $\widetilde{\kappa} = 0$).

5.2.2. **Pricing accuracy and efficiency.** We then perform multi-dimensional pricing using our method with

$$d = \{5, 10, 15, 20, 30, 40, 60, 80, 100\},\ P = 100,\ M = 10,000,$$

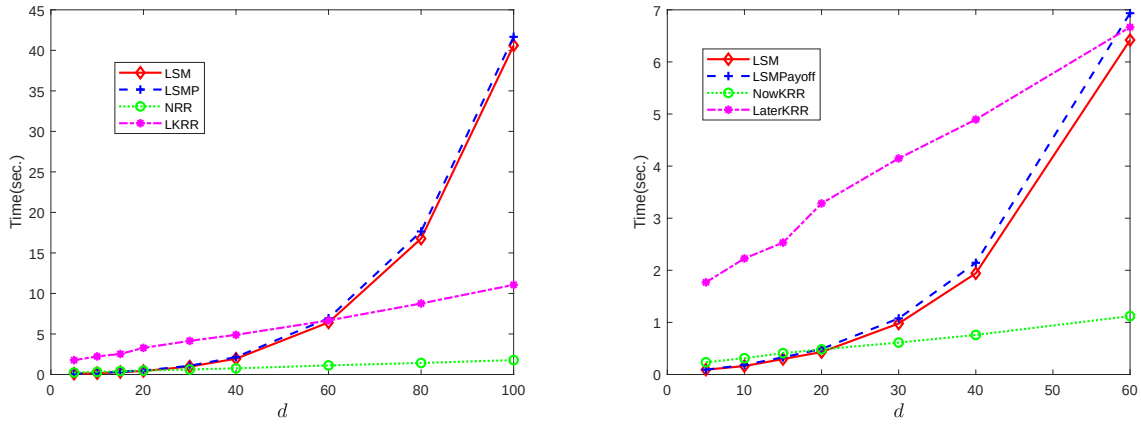
without utilizing the fact that the options can be converted to one dimension. We choose the hyperparameters of the KRR as $\lambda = 1$ and $C = d \times 10^4$ instead of a constant C to get reasonable results, and this is possibly because ρ_{ij} and ρ_{ij}^J will also change when d changes (see (56)). For the infinite sum in Proposition 2, we sum from $k = 0$ to $k = 2$ and truncate the remaining terms, which is sufficiently accurate for the numerical test cases in this section. We again set LSM with polynomial basis functions up to the second power, and LSMP with additional basis of the payoff function.

Figure 8 presents the pricing results for the various algorithms, with detailed data in Table 3. We can see that LSM and LSMP perform poorly in MJD, especially when $d \geq 30$, with the errors all exceeding 5% and even reaching about 40%. In contrast, NKRR and LKRR perform quite well for each d, with all errors below 2% for NKRR and 1.02% for LKRR. Figure 9 shows the computation time. It is quite similar as the results under GBM, while LKRR is relatively slower this time.



(a) $5 \le d \le 100$ (b) $5 \le d \le 30$ (zoomed in)

Figure 8. Pricing results under MJD for a range of dimensions.



(a) $5 \le d \le 100$ (b) $5 \le d \le 60$ (zoomed in)

Figure 9. Computation time under MJD for a range of dimensions with $P = 100$.

LSM/LSMP run a little faster than NKRR when $d < 20$. However, the computation time increases exponentially with d, and LSM/LSMP is slower than LKRR when $d > 60$. On the contrary, the computation time of NKRR/LKRR increases linearly with d.

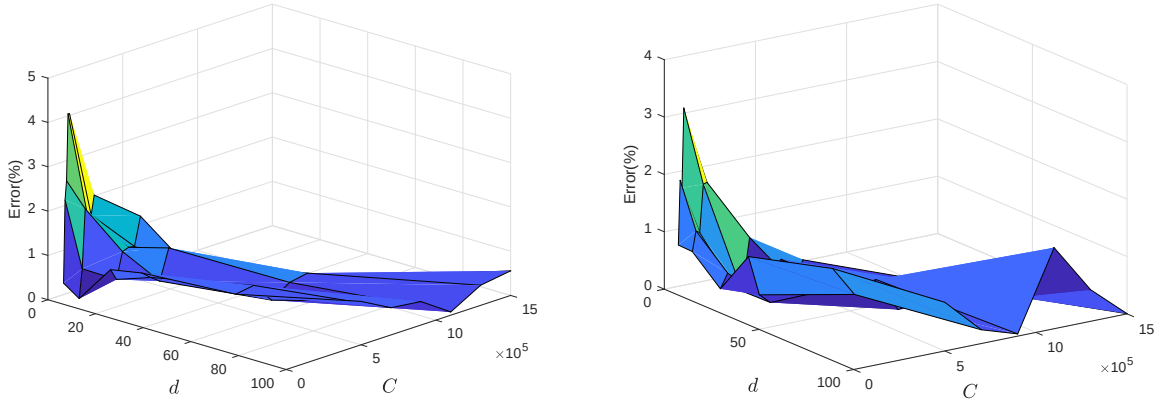
From the pricing tests above, we have learned that both NKRR and LKRR can efficiently give accurate pricing results under GBM and MJD models. NKRR runs faster and it is simple, but LKRR can provide relatively more accurate results. We can make a choice according to the practical requirements.

Table 3. Pricing results under MJD for a range of dimensions.

d	LSM	Error(%)	LSMP	Error(%)	NKRR	Error(%)	LKRR	Error(%)
5	6.883(0.079)	1.606	7.067(0.082)	1.023	6.994(0.099)	**0.010**	6.981(0.064)	0.205
10	6.962(0.104)	0.476	7.022(0.096)	**0.391**	7.027(0.084)	0.462	7.028(0.044)	0.465
15	7.042(0.048)	0.666	7.115(0.062)	1.722	7.049(0.090)	0.765	6.994(0.075)	**0.018**
20	7.174(0.061)	2.562	7.210(0.069)	3.075	7.022(0.082)	**0.380**	7.054(0.094)	0.842
30	7.379(0.058)	5.486	7.427(0.058)	6.179	7.034(0.098)	0.559	7.018(0.099)	**0.322**
40	7.661(0.067)	9.519	7.685(0.078)	9.859	7.122(0.071)	1.820	7.066(0.083)	**1.014**
60	8.271(0.058)	18.239	8.267(0.067)	18.178	7.065(0.077)	1.002	7.021(0.074)	**0.371**
80	8.939(0.089)	27.784	8.930(0.074)	27.665	7.091(0.066)	1.372	7.037(0.043)	**0.597**
100	9.651(0.092)	37.974	9.654(0.079)	38.016	7.077(0.091)	1.174	7.037(0.053)	**0.594**

NKRR: $\lambda = 1, C = d \times 10^4$; LKRR: $\lambda = 1, C = d \times 10^4$;
Benchmark price: 6.995; for other parameters see (55) and (56).

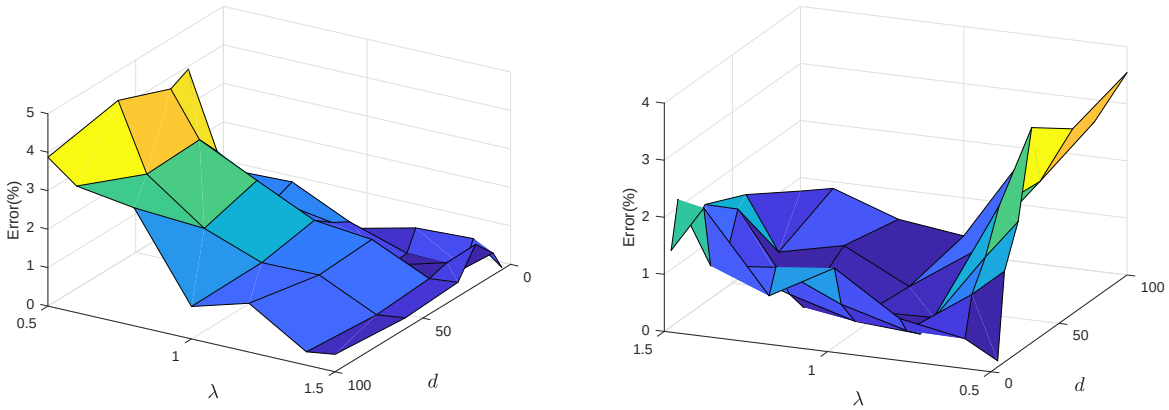
5.2.3. **Robustness test.** We can also perform a robustness test for our method under MJD. However, we only did the test from the first perspective (i.e., when C or λ changes), as there are no benchmark prices for the models with different values of the other parameters. The range of C is again set from 50% to 150%. To be specific, we test different values of d and for each d, we set $C \in [0.5d \times 10^4, 1.5d \times 10^4]$. Similarly, λ is set to vary from 0.5 to 1.5.



(a) NKRR (b) LKRR

Figure 10. Pricing results under MJD for a range of values of C and d.

Figure 10 presents the results for NKRR and LKRR. Generally, the pricing errors are within 1% for much of the range of values. Only for some part of the area for small values of d, the maximum errors attain about 4% and 3% for NKRR and LKRR, respectively. We can conclude that our method is robust with respect to the change of C. For the results for different λ's in Figure 11, we can see that $\lambda = 1$ attains local minimum error rates for NKRR, and nearly global minimum error rates for LKRR. This confirms the effectiveness of our rule for setting λ for the test cases in this section.

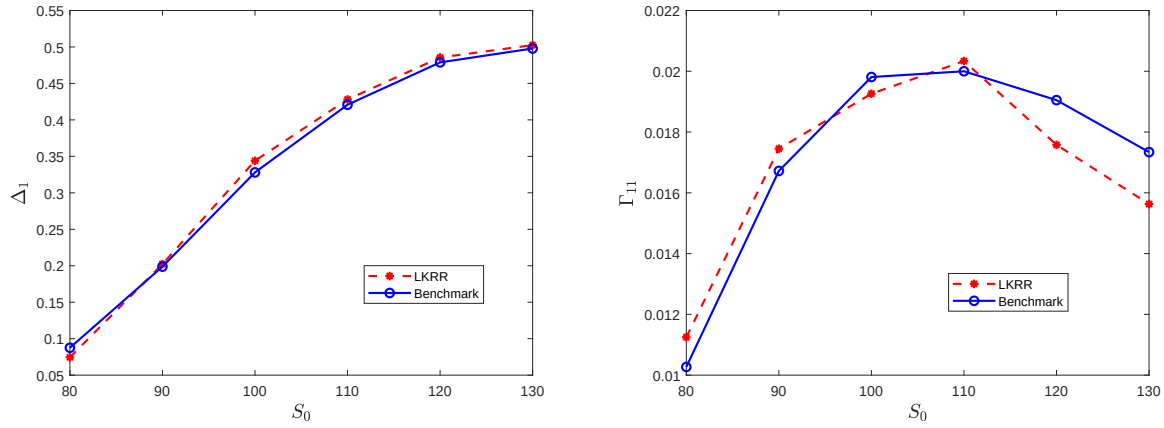


(a) NKRR (b) LKRR

Figure 11. Pricing results under MJD for a range of values of λ and d.

5.3. *Greeks of max call under GBM*

In this section we test the accuracy of the calculation of Delta and Gamma by regression-later methods with KRR under the GBM model (35). The option is a max call on two correlated assets,



(a) Δ_1 (b) Γ_{11}

Figure 12. Greeks calculation results under GBM.

and the parameters and benchmark values are from Wang and Caflisch (2010). The parameters are

$$\begin{aligned} &K = 100,\ \sigma = 0.2,\ r = 5\%,\ q = 10\%,\ \rho = 0.3,\ T = 1,\ N = 3, \\ &S_0 = \{80, 90, 100, 110, 120, 130\},\ M = 10,000,\ P = 100,\ \lambda = 1,\ C = 0.2. \end{aligned} \tag{57}$$

Figure 12 reports the result of the Greeks calculation. Our method performs well in computing the Delta. However, the accuracy of Gamma calculation is unsatisfactory, through it captures the rough shape of the curve. Our conclusion is similar to Jain and Oosterlee (2015), indicating that it is difficult to obtain accurate Gamma by the regression-later method.

6. Conclusions and future work

We have shown how to use KRR in regression-based methods for pricing high-dimensional American options by simulation. There is no basis selection step in our approach, while KRR is not a pure non-parametric method. By the kernel trick and the Gaussian kernel, KRR can expand the original input space to an infinite dimensional intrinsic space, so as to avoid selecting basis functions for the regression and obtain more accurate regression results. Our method is simple, and the numerical test results show that it is accurate, efficient and robust.

Our contribution is threefold. Firstly, we systematically introduce the main idea and theory of KRR and apply it to American option pricing for the first time. Secondly, we show how to use KRR with the Gaussian kernel in the regression-later method and give the computationally efficient formulas for estimating the continuation values and the Greeks. Thirdly, we propose to accelerate and improve the accuracy of KRR by performing local regression based on the bundling technique. We also consider the implementation aspect and carry out comprehensive numerical tests to compare the results and show the robustness of our method.

Inevitably, our method has both prons and cons, and one should pay attention to some pitfalls when applying it. First, the time complexity of KRR is $O(M^3)$, so it is not sensible to directly use KRR for a problem with large datasets. One of the solutions is to divide the samples and do local regression. Second, the selection of the kernel function is important, or else one may be unable to obtain the explicit expression of the kernel in LKRR. The optimal hyperparameters can be selected by cross-validation. However, as standard cross-validation is time consuming, it needs further consideration for problems like pricing American options by simulation, which involves regression with large samples at each time-step. Third, KRR needs to use the original training

dataset when performing the prediction. This is not memory efficient when getting lower estimation of the American option price by an early-exercise policy and newly simulated paths.

In future work, we envisage two tasks to make our method more complete. First, we plan to find a more effective way to select the Gaussian kernel parameter C and determine the optimal number of bundles P. Second, we would like to consider constructing the lower and upper price bounds for our method based on a path estimator and the dual method.

Funding

This work was supported by the Humanity and Social Science Youth foundation of the Ministry of Education of China under Grant number 16YJCZH031.

References

Andersen, L. and Broadie, M., A primal-dual simulation algorithm for pricing multidimensional American options. *Management Science*, 2004, **50**, 1222–1234.

Barraquand, J. and Martineau, D., Numerical valuation of high dimensional multivariate American securities. *Journal of Financial and Quantitative Analysis*, 1995, **30**, 383–405.

Broadie, M. and Glasserman, P., Pricing American-style securities using simulation. *Journal of Economic Dynamics and Control*, 1997, **21**, 1323–1352.

Broadie, M., Glasserman, P. *et al.*, A stochastic mesh method for pricing high-dimensional American options. *Journal of Computational Finance*, 2004, **7**, 35–72.

Broadie, M. and Yamamoto, Y., Application of the fast Gauss transform to option pricing. *Management Science*, 2003, **49**, 1071–1088.

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

Exterkate, P., Model selection in kernel ridge regression. *Computational Statistics and Data Analysis*, 2013, **68**, 1–16.

Glasserman, P., *Monte Carlo methods in financial engineering*, Vol. 53, , 2003 (Springer: New York).

Glasserman, P. and Yu, B., Simulation for American options: Regression now or regression later?. In *Monte Carlo and Quasi-Monte Carlo Methods 2002*, edited by N. H, pp. 213–226, 2004 (Springer: Berlin, Heidelberg).

Han, G.S., Kim, B.H. and Lee, J., Kernel-based Monte Carlo simulation for American option pricing. *Expert Systems with Applications*, 2009, **36**, 4431–4436.

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

Jonen, C., Efficient pricing of high-dimensional American-style derivatives: A robust regression Monte Carlo method. PhD thesis, Universität zu Köln, 2011.

Kohler, M., A review on regression-based Monte Carlo methods for pricing American options. In *Recent Developments in Applied Probability and Statistics*, edited by L. Devroye, B. Karaszen, M. Kohler and R. Korn, pp. 37–58, 2010 (Physica-Verlag HD: Heidelberg).

Kung, S.Y., *Kernel Methods and Machine Learning*, 2014 (Cambridge University Press: Cambridge).

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

Spiegeleer, J.D., Madan, D.B., Reyners, S. and Schoutens, W., Machine learning for quantitative finance: fast derivative pricing, hedging and fitting. *Quantitative Finance*, 2018, **18**, 1635–1643.

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

Wang, Y. and Caflisch, R., Pricing and hedging American-style options: a simple simulation-based approach. *Journal of Computational Finance*, 2010, **13**, 95–125.

Zhang, Y., Duchi, J.C. and Wainwright, M.J., Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *Journal of Machine Learning Research*, 2013, **30**, 592–617.

Appendix A: Proof of Lemma 1

Proof. We first write $\mathbf{X}$ as $\boldsymbol{\mu}+\mathbf{Y}$, $\mathbf{Y}\sim N(0,\boldsymbol{\Sigma})$. Then we have

$$\begin{aligned}
E\left[e^{-\frac{1}{C}\mathbf{X}^\top\mathbf{X}}\right] &= E\left[e^{-\frac{1}{C}(\boldsymbol{\mu}+\mathbf{Y})^\top(\boldsymbol{\mu}+\mathbf{Y})}\right] = e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}}E\left[e^{-\frac{1}{C}(2\boldsymbol{\mu}^\top\mathbf{Y}+\mathbf{Y}^\top\mathbf{Y})}\right] \\
&= \frac{e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}}}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}\int_{R^d} e^{-\frac{1}{C}(2\boldsymbol{\mu}^\top\mathbf{Z}+\mathbf{Z}^\top\mathbf{Z})}e^{-\frac{1}{2}\mathbf{Z}^\top\boldsymbol{\Sigma}^{-1}\mathbf{Z}}d\mathbf{Z} \\
&= \frac{e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}}}{(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{1/2}}\int_{R^d} e^{-\frac{2}{C}\boldsymbol{\mu}^\top\mathbf{Z}-\frac{1}{2}\mathbf{Z}^\top(\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1})\mathbf{Z}}d\mathbf{Z} \\
&= \frac{e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}}}{|\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1}|^{1/2}|\boldsymbol{\Sigma}|^{1/2}}\frac{|\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1}|^{1/2}}{(2\pi)^{d/2}}\int_{R^d} e^{-\frac{2}{C}\boldsymbol{\mu}^\top\mathbf{Z}-\frac{1}{2}\mathbf{Z}^\top(\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1})\mathbf{Z}}d\mathbf{Z} \\
&= \frac{1}{|\frac{2}{C}\boldsymbol{\Sigma}+\mathbf{I}|^{1/2}}e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}}E\left[e^{-\frac{2}{C}\boldsymbol{\mu}^\top\mathbf{Z}}\right],\ \mathbf{Z}\sim N\left(0,\left(\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1}\right)^{-1}\right).
\end{aligned} \tag{A1}$$

According to the formula for characteristic function of multivariate normal distribution, we have

$$\begin{aligned}
E\left[e^{-\frac{1}{C}\mathbf{X}^\top\mathbf{X}}\right] &= \frac{1}{|\frac{2}{C}\boldsymbol{\Sigma}+\mathbf{I}|^{1/2}}e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}}E\left[e^{-\frac{2}{C}\boldsymbol{\mu}^\top\mathbf{Z}}\right] \\
&= \frac{1}{|\frac{2}{C}\boldsymbol{\Sigma}+\mathbf{I}|^{1/2}}e^{-\frac{1}{C}\boldsymbol{\mu}^\top\boldsymbol{\mu}+\frac{2}{C^2}\boldsymbol{\mu}^\top(\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1})^{-1}\boldsymbol{\mu}} \\
&= \frac{1}{|\frac{2}{C}\boldsymbol{\Sigma}+\mathbf{I}|^{1/2}}e^{-\frac{2}{C^2}\boldsymbol{\mu}^\top\left[\frac{C}{2}\mathbf{I}-\left(\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1}\right)^{-1}\right]\boldsymbol{\mu}}.
\end{aligned} \tag{A2}$$

It can be verified that

$$-\frac{2}{C^2}\left[\frac{C}{2}\mathbf{I}-\left(\frac{2}{C}\mathbf{I}+\boldsymbol{\Sigma}^{-1}\right)^{-1}\right] = -\frac{1}{C}\left(\frac{2}{C}\boldsymbol{\Sigma}+\mathbf{I}\right)^{-1}. \tag{A3}$$

This concludes the proof. □

Appendix B: Proof of Proposition 1

Proof. As

$$\mathbf{X}_{i+1}-\mathbf{X}_{i+1}^{(m)}\Big|\mathbf{X}_i \sim N\left(\boldsymbol{\mu}_{i+1}-\mathbf{X}_{i+1}^{(m)},\ h\boldsymbol{\Sigma}^S\right), \tag{B1}$$

where $\boldsymbol{\mu}_{i+1}=(\mu_{i+1,1},...,\mu_{i+1,d})^\top$ and

$$\mu_{i+1,\nu} = X_{i,\nu}+(r-q_\nu-\frac{1}{2}\sigma_\nu^2)h,\ \nu=1,...,d, \tag{B2}$$

according to Lemma 1 we have

$$\begin{aligned} &E\left[K(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1})|\mathbf{X}_i\right] \\ &= \frac{1}{|\frac{2h}{C}\mathbf{\Sigma}^S + \mathbf{I}|^{1/2}} e^{-\frac{1}{C}\left(\boldsymbol{\mu}_i - \mathbf{X}_{i+1}^{(m)}\right)^\top \left(\frac{2h}{C}\mathbf{\Sigma}^S + \mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_i - \mathbf{X}_{i+1}^{(m)}\right)}. \end{aligned} \tag{B3}$$

By substituting (B3) into (34) we will get

$$\begin{aligned} \widehat{Q}_i(\mathbf{X}_i) &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} E\left[\mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1})|\mathbf{X}_i\right] \\ &= \frac{D_{i,i+1}}{|\frac{2h}{C}\mathbf{\Sigma}^S + \mathbf{I}|^{1/2}} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} e^{-\frac{1}{C}\left(\boldsymbol{\mu}_{i+1} - \mathbf{X}_{i+1}^{(m)}\right)^\top \left(\frac{2h}{C}\mathbf{\Sigma}^S + \mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_{i+1} - \mathbf{X}_{i+1}^{(m)}\right)}. \end{aligned} \tag{B4}$$

□

Appendix C: Proof of Proposition 2

Proof.

$$\begin{aligned} \widehat{Q}_i(\mathbf{X}_i) &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} E\left[\mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1})|\mathbf{X}_i\right] \\ &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} E\left[E[\mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1})|\mathbf{X}_i, \Delta N_i = k]\right] \\ &= D_{i,i+1} \sum_{k=0}^{\infty} \frac{e^{-\lambda h}(\lambda h)^k}{k!} \sum_{m=1}^{M} \widehat{\alpha}_{i+1,m} E\left[\mathcal{K}(\mathbf{X}_{i+1}^{(m)}, \mathbf{X}_{i+1})|\mathbf{X}_i, \Delta N_i = k\right], \end{aligned} \tag{C1}$$

where the inner summation can be calculated according to Lemma 1 with the corresponding normal distribution in (44). □

Appendix D: Proof of Proposition 3

Proof. We have

$$\begin{aligned} \Delta^\nu = \frac{\partial \widehat{Q}_0(\mathbf{S}_0)}{\partial S_0^\nu} &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{1,m} \frac{\partial E\left[\mathcal{K}(\mathbf{S}_1^{(m)}, \mathbf{S}_1)|\mathbf{S}_0\right]}{\partial S_0^\nu} \\ &= D_{i,i+1} \sum_{m=1}^{M} \widehat{\alpha}_{1,m} \frac{\partial E\left[\mathcal{K}(\mathbf{X}_1^{(m)}, \mathbf{X}_1)|\mathbf{X}_0\right]}{\partial X_0^\nu} \frac{1}{S_0^\nu}. \end{aligned} \tag{D1}$$

According to Proposition 1, one can obtain by straightforward manipulations

$$\frac{\partial E\left[\mathcal{K}(\mathbf{X}_1^{(m)}, \mathbf{X}_1)|\mathbf{X}_0\right]}{\partial X_0^\nu} = \gamma_{m,\nu} E\left[\mathcal{K}(\mathbf{X}_1^{(m)}, \mathbf{X}_1)|\mathbf{X}_0\right], \tag{D2}$$

where $\boldsymbol{\gamma}_m = \boldsymbol{\Sigma}^\gamma \left(\boldsymbol{\mu}_1 - \mathbf{X}_1^{(m)}\right)$ and $\boldsymbol{\Sigma}^\gamma = -\frac{2}{C}\left(\frac{2h}{C}\boldsymbol{\Sigma}^S + \mathbf{I}\right)^{-1}$. Therefore,

$$\begin{aligned}\Delta^\nu &= \frac{D_{i,i+1}}{S_0^\nu}\sum_{m=1}^{M}\widehat{\alpha}_{1,m}\gamma_{m,\nu}E\left[\mathcal{K}(\mathbf{X}_1^{(m)},\mathbf{X}_1)|\mathbf{X}_0\right] \\ &= \frac{D_{i,i+1}}{S_0^\nu|\frac{2h}{C}\boldsymbol{\Sigma}^S+\mathbf{I}|^{1/2}}\sum_{m=1}^{M}\widehat{\alpha}_{1,m}\gamma_{m,\nu}e^{-\frac{1}{C}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)^\top\left(\frac{2h}{C}\boldsymbol{\Sigma}^S+\mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)}.\end{aligned} \tag{D3}$$

Similarly,

$$\begin{aligned}\Gamma^\nu =&\frac{\partial^2\hat{Q}_0(\mathbf{S}_0)}{(\partial S_0^\nu)^2} = \frac{\partial\Delta^\nu}{\partial X_0^\nu}\frac{1}{S_0^\nu} \\ =&\frac{D_{i,i+1}}{(S_0^\nu)^2|\frac{2h}{C}\boldsymbol{\Sigma}^S+\mathbf{I}|^{1/2}} \\ &\sum_{m=1}^{M}\widehat{\alpha}_{1,m}\left(\Sigma_{\nu\nu}^\gamma+\gamma_{m,\nu}^2-\gamma_{m,\nu}\right)e^{-\frac{1}{C}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)^\top\left(\frac{2h}{C}\boldsymbol{\Sigma}^S+\mathbf{I}\right)^{-1}\left(\boldsymbol{\mu}_1-\mathbf{X}_1^{(m)}\right)},\sqrt{}\end{aligned} \tag{D4}$$

where we use the fact that $\frac{\partial\gamma_{m,\nu}}{\partial X_0^\nu} = \Sigma_{\nu\nu}^\gamma$. □