SIAM J. SCI. COMPUT.
Vol. 41, No. 1, pp. B153–B180

A NEW APPROACH FOR AMERICAN OPTION PRICING: THE DYNAMIC CHEBYSHEV METHOD*

KATHRIN GLAU†, MIRCO MAHLSTEDT‡, AND CHRISTIAN PÖTZ§

Abstract. We introduce a new method to price American options based on Chebyshev interpolation. In each step of a dynamic programming time-stepping we approximate the value function with Chebyshev polynomials. The key advantage of this approach is that it allows us to shift the model-dependent computations into an offline phase prior to the time-stepping. In the offline part a family of generalized (conditional) moments is computed by an appropriate numerical technique such as a Monte Carlo, PDE, or Fourier transform based method. Thanks to this methodological flexibility the approach applies to a large variety of models. Online, the backward induction is solved on a discrete Chebyshev grid, and no (conditional) expectations need to be computed. For each time step the method delivers a closed form approximation of the price function along with the options' delta and gamma. Moreover, the same family of (conditional) moments yield multiple outputs including the option prices for different strikes, maturities, and different payoff profiles. We provide a theoretical error analysis and find conditions that imply explicit error bounds for a variety of stock price models. Numerical experiments confirm the fast convergence of prices and sensitivities. An empirical investigation of accuracy and runtime also shows an efficiency gain compared with the least-squares Monte Carlo method introduced by Longstaff and Schwartz [*Rev. Financ. Stud.*, 14 (2001), pp. 113–147]. Moreover, we show that the proposed algorithm is flexible enough to price barrier and multivariate barrier options.

Key words. American option pricing, complexity reduction, dynamic programming, polynomial interpolation

AMS subject classifications. 91G60, 41A10

DOI. 10.1137/18M1193001

1. Introduction.

A challenging task for financial institutions is the computation of prices and sensitivities for large portfolios of derivatives such as equity options. Typically, equity options have an early exercise feature and can either be exercised at any time until maturity (American type) or at a set of predefined exercise dates (Bermudan type). Lacking explicit solutions, different numerical methods have been developed to tackle this problem. One of the first algorithms to compute American put option prices in the Black–Scholes model has been proposed by [4]. In this approach, the related partial differential inequality is solved by a finite difference scheme. A rich literature further developing the PDE approach has accrued since, including methods for jump models [22], [17], extensions to two dimensions [16], and combinations with complexity reduction techniques [15]. Besides PDE based methods a variety of other approaches have been introduced, many of which trace back to the solution of the

*Submitted to the journal's Computational Methods in Science and Engineering section June 8, 2018; accepted for publication (in revised form) November 19, 2018; published electronically February 19, 2019.

http://www.siam.org/journals/sisc/41-1/M119300.html

Funding: The second and third authors were supported by the KPMG Center of Excellence in Risk Management. This project has received funding from the European Union's Horizon 2020 research and innovation program under grant 665667.

†Ecole Polytechnique Fédérale de Lausanne, Switzerland, and School of Mathematical Sciences, Queen Mary University of London, London, E1 4NS, UK (k.glau@qmul.ac.uk).

‡Technical University of Munich, Garching-Hochbrück, Germany (mirco.mahlstedt@tum.de).

§Queen Mary University of London, London, UK, and Technical University of Munich, Germany (c.poetz@qmul.ac.uk).

optimal stopping problem by the dynamic programming principle; see, e.g., [28]. For Fourier based solution schemes we refer to [24], [10]. Simulation based approaches are of fundamental importance; the most prominent representative of this group is the least-squares Monte Carlo (LSM) approach of [23]; we refer to [13] for an overview of different Monte Carlo methods. Fourier and PDE methods typically are highly efficient; compared to simulation, however, they are less flexible toward changes in the model and particularly in the dimensionality. In order to reconcile the advantages of the PDE and Fourier approach with the flexibility of Monte Carlo simulation, we propose a new approach.

Like most approaches, we discretize the continuous time problem of pricing an American option and then solve it. Hence, we actually compute the price of a Bermudan option. It is well known that the Bermudan price converges toward the American option price and thus one can either use a high number of exercise rights or an extrapolation technique to obtain the American option price; see [12]. The pricing of Bermudan options is similar to the pricing of discretely monitored barrier options as stated in [10]. Our proposed new approach will be general enough to cover this pricing problem as well.

We consider a general dynamic programming time-stepping in discrete time. Let X_t be the underlying Markov process and the value function V_t is given by

$$\begin{aligned} V_T(x) &= g(x), \\ V_t(x) &= f\left(g(t,x), \mathbb{E}[V_{t+1}(X_{t+1})|X_t = x]\right) \end{aligned}$$

with time steps $t < t+1 < \cdots < T$ and payoff function g. The computational challenge is to compute $\mathbb{E}[V_{t+1}(X_{t+1})|X_t = x]$ for *for all time steps t and all states x*, where V_{t+1} depends on *all previous time steps*.

In order to tackle this problem, we approximate the value function in each time step by Chebyshev polynomial interpolation. We thus express the value function V_{t+1} as a finite sum of Chebyshev polynomials $T_j(x) = \cos(j \operatorname{acos}(x))$ times coefficients c_j^{t+1}. In this case, the conditional expectations become

$$\mathbb{E}[V_{t+1}(X_{t+1})|X_t = x] \approx \sum c_j^{t+1} \mathbb{E}[T_j(X_{t+1})|X_t = x] = \sum c_j^{t+1} \Gamma_{j,k} \tag{1.1}$$

with generalized moments $\Gamma_{j,k} := \mathbb{E}[T_j(X_{t+1})|X_t = x]$. The choice of Chebyshev polynomials is motivated by the promising properties of Chebyshev interpolation such as the following.

- The vector of coefficients $(c_j^{t+1})_{j=0,\ldots,N}$ is explicitly given as a linear combination of the values $V_t(x_k)$ at the Chebyshev grid points x_k. For this, (1.1) needs to be solved at the Chebyshev grid points $x = x_k$ only.
- Exponential convergence of the interpolation for analytic functions and polynomial convergence of differential functions depending on the order.
- The interpolation can be implemented in a numerically stable way.

The computation of the continuation value at a single time step coincides with the pricing of a European option. Its interpolation with Chebyshev polynomials is proposed in [11], where the method shows to be highly promising and exponential convergence is established for a large set of models and option types. Moreover, the approximation of the value function with Chebyshev polynomials has already proven to be beneficial for optimal control problems in economics; see [19] and [5].

The key advantage of our approach for American option pricing is that it collects all model-dependent computations in the generalized conditional moments $\Gamma_{j,k}$. If there is no closed-form solution, their calculation can be shifted into an offline phase

prior to the time-stepping. Depending on the underlying model a suitable numerical technique such as Monte Carlo, PDE, or Fourier transform methods can be chosen, which reveals the high flexibility of the approach. Once the generalized conditional moments $\Gamma_{j,k}$ are computed, the backward induction is solved on a discrete Chebyshev grid. This avoids any computations of conditional expectations during the time-stepping. For each time step the method delivers a closed form approximation of the price function $x \mapsto \sum c_j^t T_j(x)$ along with the option's Delta and Gamma. Since the family of generalized conditional moments $\Gamma_{j,k}$ are independent of the value function, they can be used to generate multiple outputs including the option prices for different strikes, maturities, and different payoff profiles. The structure of the method is also beneficial for the calculation of expected future exposure, which is the computational bottleneck in the computation of CVA, as investigated in [14].

The offline-online decomposition separates model and payoff yielding a modular design. We exploit this structure for a thorough error analysis and find conditions that imply explicit error bounds. They reflect the modularity by decomposing into a part stemming from the Chebyshev interpolation, from the time-stepping, and from the offline computation. Under smoothness conditions the asymptotic convergence behavior is deduced.

We perform numerical experiments using the Black–Scholes model, Merton's jump diffusion model, and the constant elasticity of variance (CEV) model as a representative of a local volatility model. For the computation of the generalized conditional moments we thus use different techniques, namely, numerical integration based on Fourier transforms and Monte Carlo simulation. Numerical experiments confirm the fast convergence of option prices along with its delta and gamma. A comprehensive comparison with the LSM reveals the potential efficiency gain of the new approach, particularly when several options on the same underlying are priced.

The rest of the article is organized as follows. We introduce the problem setting and the new method in section 2 and provide the error analysis in section 3. Section 4 discusses general traits of the implementation and section 5 presents the numerical experiments. Section 6 concludes the article, followed by an appendix with the proof of the main result.

2. The Chebyshev method for dynamic programming problems. First, we present the Bellman–Wald equation as a specific form of dynamic programming. Second, we provide the necessary notation for the Chebyshev interpolation. Then we are in a position to introduce the new approach and its application to American option pricing.

2.1. Optimal stopping and dynamic programming. Let $X = (X_t)_{t \le T}$ be a Markov process with state space $\mathbb{R}^d$ defined on the filtered probability space $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{t \ge 0}, \mathbb{P})$. Let $g : [0,T] \times \mathbb{R}^d \longrightarrow \mathbb{R}$ be a continuous function with $\mathbb{E}[\sup_{0 \le t \le T} |g(t, X_t)|] < \infty$. Then

$$V(t,x) := \sup_{t \le \tau \le T} \mathbb{E}\left[g(\tau, X_\tau) | X_t = x\right] \qquad \forall (t,x) \in [0,T] \times \mathbb{R}^d$$

over all stopping times τ; see (2.2.2′) in [28]. In discrete time, the optimal stopping problems can be solved with dynamic programming.

Namely, with time-stepping $t = t_0 < \cdots < t_n = T$ the solution of the optimal stopping problem can be calculated via backward induction

$$\begin{aligned} V_T(x) &= g(T,x), \\ V_{t_u}(x) &= \max\left(g(t_u,x), \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}}) | X_{t_u} = x]\right). \end{aligned}$$

Note that n refers to the number of time steps between t and T. For notational convenience, we indicate the value function at each time step with subscript t_u to directly refer to the time step t_u. For a detailed overview of optimal control problems in discrete time we refer to [28].

2.2. Chebyshev polynomial interpolation. The univariate Chebyshev polynomial interpolation as described in detail in [33] has a tensor based extension to the multivariate case; see, e.g., [31]. Usually, the Chebyshev interpolation is defined for a function on a $[-1,1]^D$ domain. For an arbitrary hyperrectangular $\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_D, \overline{x}_D]$, we introduce a linear transformation $\tau_{\mathcal{X}} : [-1,1]^D \to \mathcal{X}$ componentwise defined by

$$\tau_{\mathcal{X}}(z_i) = \overline{x}_i + 0.5(\underline{x}_i - \overline{x}_i)(1 - z_i). \tag{2.1}$$

Let $\overline{N} := (N_1, \ldots, N_D)$ with $N_i \in \mathbb{N}_0$ for $i = 1, \ldots, D$. We define the index set

$$\mathcal{J} := \{j \in \mathbb{N}^D : 1 \le j_i \le N_D \text{ for } i = 1, \ldots, d\}.$$

The Chebyshev polynomials are defined for $z \in [-1,1]^D$ and $j \in \mathcal{J}$ by

$$T_j(z) = \prod_{i=1}^{D} T_{j_i}(z_i), \quad T_{j_i}(z_i) = \cos(j_i \cdot \operatorname{acos}(z_i)),$$

and the jth Chebyshev polynomial on $\mathcal{X}$ as $p_j(x) = T_j(\tau_{\mathcal{X}}^{-1}(x))1_{\mathcal{X}}(x)$. The Chebyshev points are given by

$$z^k = (z_{k_1}, \ldots, z_{k_D}),\ z_{k_i} = \cos\left(\pi \frac{k_i}{N_i}\right) \text{ for } k_i = 0, \ldots, N_i \text{ and } i = 1, \ldots, D$$

and the transformed Chebyshev points by $x^k = \tau_{\mathcal{X}}(z^k)$. The Chebyshev interpolation of a function $f : \mathcal{X} \to \mathbb{R}$ with $\prod_{i=1}^{D}(N_i + 1)$ summands can be written as a sum of Chebyshev polynomials

$$I_{\overline{N}}(f)(x) = \sum_{j \in \mathcal{J}} c_j T_j(\tau_{\mathcal{X}}^{-1}(x)) = \sum_{j \in \mathcal{J}} c_j p_j(x) \qquad \text{for} \quad x \in \mathcal{X} \tag{2.2}$$

with coefficients c_j for $j \in \mathcal{J}$

$$c_j = \left(\prod_{i=1}^{D} \frac{2^{\mathbb{1}_{\{0 < j_i < N_i\}}}}{N_i}\right) {\sum_{k \in \mathcal{J}}}'' f(x^k) T_j(z^k), \tag{2.3}$$

where $\sum''$ indicates the summand is multiplied by $1/2$ if $k_i = 0$ or $k_i = N_i$.

2.3. The dynamic Chebyshev method. In this section, we present the new approach to solve a dynamic programming problem (DPP) via backward induction using Chebyshev polynomial interpolation.

DEFINITION 2.1. *We consider a DPP with value function*

$$V_T(x) = g(T, x), \tag{2.4}$$

$$V_{t_u}(x) = f\left(g(t_u, x), \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x]\right), \tag{2.5}$$

where $t = t_0 < \cdots < t_n = T$ and $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is Lipschitz continuous with constant L_f.

At the initial time $T = t_n$, we apply Chebyshev interpolation to the function $g(T, x)$, i.e., for $x \in \mathcal{X}$,

$$V_T(x) = g(T, x) \approx \sum_{j \in \mathcal{J}} c_j(T) p_j(x) =: \widehat{V}_T(x).$$

At the first time step t_{n-1}, the derivation of the conditional expectation $\mathbb{E}[g(t_n, X_{t_n})|X_{t_{n-1}} = x]$ is replaced by $\mathbb{E}[\sum_j c_j(t_n) p_j(X_{t_n})|X_{t_{n-1}} = x]$ yielding

$$\begin{aligned} V_{t_{n-1}}(x) &= f\left(g(t_{n-1}, x), \mathbb{E}[V_{t_n}(X_{t_n})|X_{t_{n-1}} = x]\right) \\ &\approx f\left(g(t_{n-1}, x), \mathbb{E}\left[\sum_{j \in \mathcal{J}} c_j(t_n) p_j(X_{t_n}) \middle| X_{t_{n-1}} = x\right]\right) \\ &= f\left(g(t_{n-1}, x), \sum_{j \in \mathcal{J}} c_j(t_n) \mathbb{E}\big[p_j(X_{t_n})\big|X_{t_{n-1}} = x\big]\right). \end{aligned}$$

At time step t_{n-1} the value function $V_{t_{n-1}}$ needs only to be evaluated at the specific Chebyshev nodes. Hence, denoting with $x^k = (x_{k_1}, \ldots, x_{k_D})$ the Chebyshev nodes, it suffices to evaluate

$$V_{t_{n-1}}(x^k) \approx f\left(g(t_{n-1}, x^k), \sum_{j \in \mathcal{J}} c_j(t_n) \mathbb{E}\big[p_j(X_{t_n})\big|X_{t_{n-1}} = x^k\big]\right) =: \widehat{V}_{t_{n-1}}(x^k). \tag{2.6}$$

A linear transformation of $(\widehat{V}_{t_{n-1}}(x^k))_{k \in \mathcal{J}}$ yields the Chebyshev coefficients according to (2.3), which determines the Chebyshev interpolation $\widehat{V}_{t_{n-1}} = \sum_j c_j(t_{n-1}) p_j$. We apply this procedure iteratively as described in detail in Algorithm 2.1.

The stochastic part is gathered in the expectations of the Chebyshev polynomials conditioned on the Chebyshev nodes, i.e., $\Gamma_{j,k}(t_u) = \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x^k]$. Moreover, if an equidistant time-stepping is applied the computation can be further simplified. If for the underlying stochastic process

$$\Gamma_{j,k}(t_u) = \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x^k] = \mathbb{E}[p_j(X_{t_1})|X_{t_0} = x^k] =: \Gamma_{j,k} \tag{2.7}$$

for $u = 0, \ldots, n-1$, then the conditional expectations need to be computed only for one time step; see Algorithm 2.2. One can precompute these conditional expectations, and thus the method allows for an offline/online decomposition.

3. Error analysis. In this section we analyze the error of Algorithm 2.1, i.e.,

$$\varepsilon_{t_u} := \max_{x \in \mathcal{X}} |V_{t_u}(x) - \widehat{V}_{t_u}(x)|. \tag{3.1}$$

Two different error sources occur at t_u, the classical interpolation error of the Chebyshev interpolation and a distortion error at the nodal points. With distortion, we refer to the computational noise that comes from the fact that we do not observe the correct function values $V_{t_u}(x^k)$ at the nodal points but distorted values $\widehat{V}_{t_u}(x^k)$. We call the error induced from this noise distortion error. Later on in this section the distortion error will be discussed in more detail. The behavior of the interpolation error depends on the regularity of the value function. Here, we assume analyticity of the value function. The concept can be extended to further cases such as assuming differentiability or piecewise analyticity. The latter is discussed in preliminary form

Algorithm 2.1. Dynamic Chebyshev algorithm.

Require: $\overline{N} \in \mathbb{N}^D$, $\mathcal{X} = [\underline{x}_1, \overline{x}_1] \times \ldots \times [\underline{x}_D, \overline{x}_D]$, $0 = t_0, \ldots, t_n = T$

1: Determine index set $\mathcal{J}$ and nodal points $x^k = (x_{k_1}, \ldots, x_{k_D})$

2: **Precomputation step:**

3: For all $j, k \in \mathcal{J}$ and all $t_u,\ u = 0, \ldots, n-1$

4: Compute $\Gamma_{j,k}(t_u) = \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x^k]$

5: **Time** T

6: $\widehat{V}_T(x^k) = g(T, x^k),\ k \in \mathcal{J}$, derive

7: $c_j(T) = D_{\overline{N}}(j) \sum_{k \in \mathcal{J}}{}'' \widehat{V}_T(x^k) T_j(z^k)$

8: Obtain Chebyshev interpolation $\widehat{V}_T(x) = \sum_{j \in \mathcal{J}} c_j(T) p_j(x)$ of $V_T(x)$

9: **Iterative time-stepping** from $t_{u+1} \to t_u,\ u = n-1, \ldots, 1$

10: Given Chebyshev interpolation of $\widehat{V}_{t_{u+1}}(x) = \sum_{j \in \mathcal{J}} c_j(t_{u+1}) p_j(x)$

11: Derivation of $\widehat{V}_{t_u}(x^k),\ k \in \mathcal{J}$ at the nodal points

12: $\widehat{V}_{t_u}(x^k) = f(g(t_u, x^k), \sum_{j \in \mathcal{J}} c_j(t_{u+1}) \Gamma_{j,k}(t_u))$

13: Derive $c_j(t_u) = D_{\overline{N}}(j) \sum_{k \in \mathcal{J}}{}'' \widehat{V}_{t_u}(x^k) T_j(z^k)$

14: Obtain Chebyshev interpolation $\widehat{V}_{t_u}(x) = \sum_{j \in \mathcal{J}} c_j(t_u) p_j(x)$ of $V_{t_t}(x)$

15: **Deriving the solution at** $t = 0$

16: $\widehat{V}_0(x) = \sum_{j \in \mathcal{J}} c_j(0) p_j(x)$

Algorithm 2.2. Simplified dynamic Chebyshev algorithm.

Require: Time steps $0 = t_1, \ldots, t_n = T$ with $\Delta t := t_u - t_{u-1}$

1: Replace in Algorithm 2.1 lines 2–4 with:

2: **Precomputation step:**

3: Compute $\Gamma_{j,k} = \mathbb{E}[p_j(X_{\Delta t})|X_0 = x^k]$ for all $j, k \in \mathcal{J}$

in [25, section 5.3] and is further investigated in a follow-up paper. Hence, we need a convergence result for the Chebyshev interpolation which incorporates a distortion error at the nodal points.

First, we introduce the required notation. A *Bernstein ellipse* $\mathcal{B}([-1,1], \varrho)$ with $\varrho > 1$ is defined as the open region in the complex plane bounded by an ellipse with foci ± 1 and semiminor and semimajor axis lengths summing to ϱ. We define a *generalized Bernstein ellipse* $\mathcal{B}(\mathcal{X}, \varrho)$ around the hyperrectangle $\mathcal{X}$ with parameter vector $\varrho \in (1, \infty)^D$ as

$$\mathcal{B}(\mathcal{X}, \varrho) := \mathcal{B}([\underline{x}_1, \overline{x}_1], \varrho_1) \times \cdots \times \mathcal{B}([\underline{x}_D, \overline{x}_D], \varrho_D)$$

with $\mathcal{B}([\underline{x}, \overline{x}], \varrho) := \tau_{[\underline{x}, \overline{x}]} \circ \mathcal{B}([-1,1], \varrho)$, where for $x \in \mathbb{C}$ we have the transform $\tau_{[\underline{x}, \overline{x}]}\big(\Re(x)\big) := \overline{x} + \frac{\underline{x} - \overline{x}}{2}\big(1 - \Re(x)\big)$ and $\tau_{[\underline{x}, \overline{x}]}\big(\Im(x)\big) := \frac{\overline{x} - \underline{x}}{2}\Im(x)$, where the sets $\mathcal{B}([-1,1], \varrho_i)$ are Bernstein ellipses for $i = 1, \ldots, D$.

PROPOSITION 3.1. *Let $\mathcal{X} \ni x \mapsto f(x)$ be a real-valued function with an analytic extension to some generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}, \varrho)$ for $\varrho \in (1, \infty)^D$ with $\sup_{x \in \mathcal{B}(\mathcal{X}, \varrho)} |f(x)| \le b$. Assume distorted values $f^\varepsilon(x^k) = f(x^k) + \varepsilon(x^k)$ with $|\varepsilon(x^k)| \le \overline{\varepsilon}$ at all nodes x^k. Then*

$$\max_{x \in \mathcal{X}} \big| f(x) - I_{\overline{N}}(f^\varepsilon)(x) \big| \le \varepsilon_{int}(\varrho, N, D, B) + \overline{\varepsilon} \Lambda_{\overline{N}}$$

with

$$\varepsilon_{int}(\varrho, N, D, B) := 2^{\frac{D}{2}+1} \cdot B \cdot \left(\sum_{i=1}^{D} \varrho_i^{-2N_i} \prod_{j=1}^{D} \frac{1}{1-\varrho_j^{-2}} \right)^{\frac{1}{2}} \tag{3.2}$$

and Lebesgue constant $\Lambda_{\overline{N}} \leq \prod_{i=1}^{D} \left(\frac{2}{\pi} \log(N_i + 1) + 1 \right)$.

Proof. Using the linearity of the interpolation operator we obtain for the Chebyshev interpolation of f^ε with $f^\varepsilon(x^k) = f(x^k) + \varepsilon(x^k)$ that

$$I_{\overline{N}}(f^\varepsilon)(x) = I_{\overline{N}}(f)(x) + I_{\overline{N}}(\varepsilon)(x).$$

The tensor-based multivariate Chebyshev interpolation $I_{\overline{N}}(\varepsilon)$ can be written in Lagrange form

$$I_{\overline{N}}(\varepsilon)(x) = \sum_{j \in \mathcal{J}} \varepsilon(x^j) \lambda^j(x) \qquad \text{with} \quad \lambda^j(x) = \prod_{i=1}^{D} \ell_{j_i}(\tau^{-1}_{[\underline{x}_i, \overline{x}_i]}(x_i)),$$

where $\ell_{j_i}(z) = \prod_{k \neq j_i} \frac{z - z_k}{z_{j_i} - z_k}$ is the j_ith Lagrange polynomial. This yields

$$\max_{x \in \mathcal{X}} |I_N(\varepsilon)(x)| = \max_{x \in \mathcal{X}} \left| \sum_{j \in \mathcal{J}} \varepsilon(x^j) \lambda^j(x) \right| \leq \overline{\varepsilon} \max_{x \in \mathcal{X}} \sum_{j \in \mathcal{J}} |\lambda^j(x)| =: \overline{\varepsilon} \Lambda_{\overline{N}}.$$

The term $\Lambda_{\overline{N}}$ is the Lebesgue constant of the (multivariate) Chebyshev nodes which is given by

$$\Lambda_{\overline{N}} = \max_{x \in \mathcal{X}} \sum_{j \in \mathcal{J}} \left| \lambda^j(x) \right| = \max_{x \in \mathcal{X}} \sum_{j \in \mathcal{J}} \prod_{i=1}^{D} \left| \ell_{j_i}(x_i) \right| = \prod_{i=1}^{D} \max_{x_i \in [\underline{x}_i, \overline{x}_i]} \sum_{j_i=0}^{N_i} \left| \ell_{j_i}\left(\tau^{-1}_{[\underline{x}_i, \overline{x}_i]}(x_i) \right) \right|.$$

Since $\max_{x_i \in [\underline{x}_i, \overline{x}_i]} \sum_{j_i=0}^{N_i} |\ell_{j_i}(\tau^{-1}_{[\underline{x}_i, \overline{x}_i]}(x_i))| = \max_{z \in [-1,1]} \sum_{j_i=0}^{N_i} |\ell_{j_i}(z)| = \Lambda_{N_i}$, which is the Lebesgue constant of the univariate Chebyshev interpolation, we have $\Lambda_{\overline{N}} = \prod_{i=1}^{D} \Lambda_{N_i}$. From [33, Theorem 15.2] we obtain for the univariate Chebyshev interpolation $\Lambda_N \leq \frac{2}{\pi} \log(N+1) + 1$ and hence

$$\Lambda_{\overline{N}} \leq \prod_{i=1}^{D} \Big(\frac{2}{\pi} \log(N_i + 1) + 1 \Big). \tag{3.3}$$

For the distorted Chebyshev interpolation it holds that

$$\left| f(x) - I_{\overline{N}}(f^\varepsilon)(x) \right| \leq \left| f(x) - I_{\overline{N}}(f)(x) \right| + \left| I_{\overline{N}}(\varepsilon)(x) \right|.$$

Therefore, the proposition follows directly from (3.3) and [31]. ∎

We use this result to investigate the error of the dynamic Chebyshev method. First, we introduce the following assumption.

Assumption 3.2. We assume $\mathcal{X} \ni x \mapsto V_{t_u}(x)$ is a real valued function that has an analytic extension to a generalized Bernstein ellipse $\mathcal{B}(\mathcal{X}, \varrho_{t_u})$ with $\varrho_{t_u} \in (1, \infty)^D$ and $\sup_{x \in \mathcal{B}(\mathcal{X}, \varrho_{t_u})} |V_{t_u}(x)| \leq B_{t_u}$ for $u = 1, \ldots, n$.

Proposition 3.5 provides conditions on the process X and the functions f and g that guaranty Assumptions 3.2. Under this assumptions, we can apply Proposition 3.1 to obtain an error bound for the dynamic Chebyshev method at each time step. This error bound has a recursive structure, since the values of V_{t_u} depend on the conditional expectation of $V_{t_{u+1}}$. The interpolation error of the final time step is of form (3.2). At any other time step t_u an additional distortion error by approximating the function values at the nodal points by

$$V_{t_u}(x^k) \approx f\left(g(t_u, x^k), \sum_{j\in\mathcal{J}} c_j(t_{u+1})\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x^k]\right) = \widehat{V}_{t_u}(x^k)$$

comes into play. Proposition 3.1 yields

$$\varepsilon_{t_u} := \max_{x\in\mathcal{X}} |V_{t_u}(x) - \widehat{V}_{t_u}(x)| \leq \varepsilon_{int}(\varrho_{t_u}, N, D, B_{t_u}) + \Lambda_{\overline{N}} F_{t_u},$$

where $F_{t_u} := \max_{j\in\mathcal{J}} |V_{t_u}(x_j) - \widehat{V}_{t_u}(x_j)|$. The term F_{t_u} depends on the function f and the interpolation error at the previous time step t_{u+1}.

Moreover, two additional error sources can influence the error bound. If there is no closed-form solution for the generalized moments $\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x^k]$ a numerical technique, e.g., numerical quadrature or Monte Carlo methods, introduces an additional error. The former is typically deterministic and bounded whereas the latter is stochastic. In order to incorporate this error in the following error analysis we introduce some additional notation. The conditional expectation can be seen as a linear operator which operates on the vector space of all continuous functions $\mathcal{C}(\mathbb{R}^D)$ with finite L^∞-norm

$$\Gamma^k_{t_u} : \mathcal{C}(\mathbb{R}^D) \to \mathbb{R} \quad \text{with} \quad \Gamma^k_{t_u}(f) := \mathbb{E}[f(X_{t_{u+1}})|X_{t_u} = x^k].$$

Define the subspace of all D variate polynomials $\mathcal{P}_{\overline{N}}(\mathcal{X}) := \text{span}\{p_j,\ j \in \mathcal{J}\}$ equipped with the L^∞-norm. We assume the operator $\Gamma^k_{t_u}$ is approximated by a linear operator $\widehat{\Gamma}^k_{t_u} : \mathcal{P}_{\overline{N}}(\mathcal{X}) \to \mathbb{R}$ on $\mathcal{P}_{\overline{N}}(\mathcal{X})$ which fullfills one of the two following conditions. For all $u = 0, \ldots, n$ the approximation is either deterministic and the error is bounded by a constant $\overline{\delta}$,

$$\text{(GM)} \qquad ||\Gamma^k_{t_u} - \widehat{\Gamma}^k_{t_u}||_{op} := \sup_{\substack{p\in\mathcal{P}_{\overline{N}} \\ ||p||=1}} \left|\Gamma^k_{t_u}(p) - \widehat{\Gamma}^k_{t_u}(p)\right| \leq \overline{\delta} \quad \forall k \in \mathcal{J},$$

or the approximation is stochastic and uses M samples of the underlying process and the polynomials p may have stochastic coefficients. In this case we assume the error bound

$$\text{(GM*)} \qquad ||\Gamma^k_{t_u} - \widehat{\Gamma}^k_{t_u}||_{op} := \sup_{\substack{p\in\mathcal{P}_{\overline{N}} \\ ||p||^\star_\infty=1}} \mathbb{E}\left[\left|\Gamma^k_{t_u}(p) - \widehat{\Gamma}^k_{t_u}(p)\right|\right] \leq \delta^\star(M) \quad \forall k \in \mathcal{J}$$

with norm $||p||^\star_\infty = \max_{x\in\mathcal{X}} \mathbb{E}[|p(x)|]$. In order to incorporate stochasticity of $\widehat{V}_{t_u}(x)$, we replace (3.1) by

$$\varepsilon_{t_{u+1}} := \max_{x\in\mathcal{X}} \mathbb{E}\left[\left|V_{t_u}(x) - \widehat{V}_{t_u}(x)\right|\right]. \tag{3.4}$$

Note that in the deterministic case (3.1) and (3.4) coincide. Additionally, a truncation error is introduced by restricting to a compact interpolation domain $\mathcal{X}$. We assume

that the conditional expectation of the value function outside this set is bounded by a constant

$$\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})\mathbb{1}_{\mathbb{R}^D\setminus\mathcal{X}}|X_{t_u}=x^k]\le\varepsilon_{tr}. \tag{TR}$$

The following theorem provides an error bound for the dynamic Chebyshev method.

THEOREM 3.3. *Let the DPP be given as in Definition* 2.1. *Assume the regularity Assumptions* 3.2 *hold and the boundedness of the truncation error* (TR). *Then we have*

$$\varepsilon_{t_u}\le\sum_{j=u}^{n}C^{j-u}\varepsilon_{int}^{j}\ +\ \Lambda_{\overline{N}}L_f\sum_{j=u+1}^{n}C^{j-(u+1)}(\varepsilon_{tr}+\varepsilon_{gm}\overline{V}_j) \tag{3.5}$$

with $\varepsilon_{gm}=\overline{\delta}$ *if assumption* (GM) *holds and* $\varepsilon_{gm}=\delta^\star(M)$ *if assumption* (GM*) *holds and* $C=\Lambda_{\overline{N}}L_f(1+\varepsilon_{gm})$, $\overline{V}_j=\max_{x\in\mathcal{X}}|V_{t_j}(x)|$, *and* $\varepsilon_{int}^{j}=\varepsilon_{int}(\varrho_{t_j},N,D,B_{t_j})$.

Proof. The proof of the theorem can be found in the appendix. ∎

The following corollary provides a simplified version of the error bound (3.5) presenting its decomposition into three different error sources (interpolation error ε_{int}, truncation error ε_{tr}, and the error from the numerical computation of the generalized moments ε_{gm}).

COROLLARY 3.4. *Let the setting be as in Theorem* 3.3. *Then the error is bounded by*

$$\varepsilon_{t_u}\le\left(\varepsilon_{int}(\underline{\varrho},N,D,\overline{B})+\varepsilon_{tr}+\varepsilon_{gm}\overline{V}\right)\tilde{C}^{n+1-u} \tag{3.6}$$

with $\tilde{C}=\max\{2,C\}$, $\underline{\varrho}=\min_{1\le u\le n}\varrho_{t_u}$, $\overline{B}=\max_{1\le u\le n}B_{t_u}$, $\overline{V}=\max_{u\le j\le n}\overline{V}_j$.

Moreover, if $\varepsilon_{tr}=0$, $L_f\le 1$, *and* $N=N_i$, $i=1,\dots,D$, *the error bound can be simplified further. Under* (GM*) $\delta^\star(M)\le c/\sqrt{M}$, $c>0$ *yields*

$$\varepsilon_{t_u}\le\tilde{c}_1\varrho^{-N}\log(N)^{D\,n}\ +\ \tilde{c}_2\log(N)^{D\,n}M^{-0.5}$$

for some constants $\tilde{c}_1,\tilde{c}_2>0$. *Under* (GM) *the term* $M^{-0.5}$ *is replaced by* $\overline{\delta}$.

Proof. Assuming $C>2$ and using the geometric series, the first term in the error bound (3.5) can be rewritten as

$$\sum_{j=u}^{n}C^{j-u}\varepsilon_{int}^{j}\le\overline{\varepsilon}_{int}\sum_{j=u}^{n}C^{j-u}=\overline{\varepsilon}_{int}\sum_{k=0}^{n-u}C^{k}=\overline{\varepsilon}_{int}\left(\frac{1-C^{n+1-u}}{1-C}\right)\le\overline{\varepsilon}_{int}\,C^{n+1-u},$$

where $\overline{\varepsilon}_{int}=\max_j\varepsilon_{int}^{j}=\max_j\varepsilon_{int}(\varrho_{t_j},N,D,B_{t_j})\le\varepsilon_{int}(\underline{\varrho},N,D,\overline{B})$ for $\underline{\varrho}=\min_{1\le u\le n}\varrho_u$ and $\overline{B}=\max_{1\le u\le n}B_{t_u}$. For $C\le 2$ the sum is bounded by $\overline{\varepsilon}_{int}\,2^{n+1-u}$. Similarly, we obtain for the second term in the error bound (3.5) with $\beta=(\varepsilon_{tr}+\varepsilon_{gm}\overline{V}_j)$

$$\Lambda_{\overline{N}}L_f\sum_{j=u+1}^{n}C^{j-(u+1)}\beta_j\le\Lambda_{\overline{N}}L_f\,\overline{\beta}\sum_{k=0}^{n-(u+1)}C^{k}\le\Lambda_{\overline{N}}L_f\,\overline{\beta}C^{n-u}\le\overline{\beta}C^{n+1-u},$$

where $\overline{\beta}=\max_j\beta_j$. Moreover, we used $\Lambda_{\overline{N}}L_f\le\Lambda_{\overline{N}}L_f(1+\varepsilon_{gm})=C$ in the last step. Thus, we obtain the following error bound (3.5):

$$\varepsilon_{t_u}\le(\overline{\varepsilon}_{int}\ +\ \overline{\beta})\,\tilde{C}^{n+1-u}=\left(\varepsilon_{int}(\underline{\varrho},N,D,\overline{B})+\varepsilon_{tr}+\varepsilon_{gm}\overline{V}\right)\tilde{C}^{n+1-u},$$

where $\tilde{C}=\max\{2,C\}$ and $\overline{V}=\max_j\overline{V}_j$, which shows (3.6).

Furthermore, using the definition of the error bound (3.2) and $N = N_i$, $i = 1, \dots, D$, we conclude that $\varepsilon_{int}(\underline{\varrho}, N, D, \overline{B}) \leq c_1 \varrho^{-N}$ for a constant $c_1 > 0$. For the Lebesgue constant of the Chebyshev interpolation there exists a constant $c_2 > 0$ such that

$$\Lambda_{\overline{N}} \leq \prod_{i=1}^{D} \Big(\frac{2}{\pi} \log(N+1) + 1 \Big) \leq \prod_{i=1}^{D} \Big(\frac{4}{\pi} + 1 \Big) \log(N) \leq c_2 \log(N)^D.$$

Under (GM*), $\delta^\star(M) \leq c/\sqrt{M}$, $c > 0$, yields with $\varepsilon_{tr} = 0$, $L_f \leq 1$,

$$\begin{aligned} \varepsilon_{t_u} &\leq \left(\varepsilon_{int}(\underline{\varrho}, N, D, \overline{B}) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}\right) \left(\Lambda_{\overline{N}} L_f (1 + \varepsilon_{gm})\right)^{n+1-u} \\ &\leq \left(c_1 \varrho^{-N} + c\overline{V} M^{-0.5}\right) \left(c_2 \log(N)^D (1 + cM^{-0.5})\right)^n \\ &\leq \tilde{c}_1 \varrho^{-N} \log(N)^{D\,n} \;+\; \tilde{c}_2 \log(N)^{D\,n} M^{-0.5} \end{aligned}$$

and this converges toward zero for $N \to \infty$ if $\sqrt{M} > \log(N)^{D\,n}$. If (GM) holds, we have $\varepsilon_{gm} = \overline{\delta}$ and the term $M^{-0.5}$ is replaced by $\overline{\delta}$. ∎

The following proposition provides conditions under which the value function has an analytic extension to some generalized Bernstein ellipse and Assumption 3.2 holds.

PROPOSITION 3.5. *Consider a DPP as defined in* (2.4) *and* (2.5) *with equidistant time-stepping and* $g_t(x) := g(t,x)$. *Let* $X = (X_t)_{0 \leq t \leq T}$ *be a Markov process with stationary increments. Assume* $e^{\langle \eta, \cdot \rangle} g_{t_u}(\cdot) \in L^1(\mathbb{R}^D)$ *for some* $\eta \in \mathbb{R}^D$ *and* g_{t_u} *has an analytic extension to the generalized Bernstein ellipse* $\mathcal{B}(\mathcal{X}, \varrho_g)$ *for* $u = 0, \dots, n$. *Furthermore, assume* $f : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ *has an analytic extension to* $\mathbb{C}^2$. *If*

(i) *the characteristic function* φ^x *of* $X_{\Delta t}$ *with* $X_0 = x$ *is in* $L^1(\mathbb{R}^D)$ *for every* $x \in \mathcal{X}$,

(ii) *for every* $z \in \mathbb{R}^D$ *the mapping* $x \mapsto \varphi^x(z - i\eta)$ *has an analytic extension to* $B(\mathcal{X}, \varrho_\varphi)$ *and there are constants* $\alpha \in (1, 2]$ *and* $c_1, c_2 > 0$ *such that* $\sup_{x \in B(\mathcal{X}, \varrho_\varphi)} |\varphi^x(z)| \leq c_1 e^{-c_2 |z|^\alpha}$ *for all* $z \in \mathbb{R}^D$,

then the value function $x \mapsto V_{t_u}(x)$ *of the DPP has an analytic extension to* $B(\mathcal{X}, \varrho)$ *with* $\varrho = \varrho_g$.

Proof. At T the value function $x \mapsto V_T(x)$ is analytic since $V_T(x) = g_T(x)$ and g_T has an analytic extension by assumption. Moreover, $e^{\langle \eta, \cdot \rangle} g_T(\cdot) \in L^1(\mathbb{R}^D)$ for some $\eta \in \mathbb{R}^D$. We assume $e^{\langle \eta, \cdot \rangle} V_{t_{u+1}}(\cdot) \in L^1(\mathbb{R}^D)$ and $V_{t_{u+1}}$ has an analytic extension to $B(\mathcal{X}, \varrho)$. Then the function

$$x \mapsto V_{t_u}(x) = f\left(g_{t_u}(x), \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}}) | X_{t_u} = x]\right)$$

is analytic if $x \mapsto \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}}) | X_{t_u} = x] = \mathbb{E}[V_{t_{u+1}}(X^x_{\Delta t})]$ has an analytic extension. From [11, Conditions 3.1] we obtain conditions (A1)–(A4) under which a function of the form $(p^1, p^2) \mapsto \mathbb{E}[f^{p^1}(X^{p^2})]$ is analytic. In our case we only have the parameter $p^2 = x$ and so $X^{p^2} = X^x_{\Delta t}$. Condition (A1) is satisfied since $e^{\langle \eta, \cdot \rangle} V_{t_{u+1}}(\cdot) \in L^1(\mathbb{R}^D)$ and for (A2) we have to verify that $|\widehat{V}_{t_{u+1}}(-z - i\eta)| \leq c_1 e^{c_2 |z|}$ for constants $c_1, c_2 > 0$.

$$\begin{aligned} |\widehat{V}_{t_{u+1}}(-z - i\eta)| &= \left| \int_{\mathbb{R}^D} e^{i\langle y, -z - i\eta \rangle} V_{t_{u+1}} \mathrm{d}y \right| \\ &\leq \int_{\mathbb{R}^D} |e^{-i\langle y, z \rangle}| \left| e^{\langle y, \eta \rangle} V_{t_{u+1}}(y) \right| \mathrm{d}y \\ &\leq ||e^{\langle \eta, \cdot \rangle} V_{t_{u+1}}(\cdot)||_{L^1} \end{aligned}$$

and thus (A2) holds. The remaining conditions (A3)–(A4) are equivalent to our conditions (i)–(ii) and [11, Theorem 3.2] yields the analyticity of $x \mapsto \mathbb{E}[V_{t_{u+1}}(X^x_{\Delta t})]$ on the Bernstein ellipse $\mathcal{B}(\mathcal{X}, \varrho_\varphi)$. Hence, $x \mapsto V_{t_u}(x)$ is a composition of analytic functions and therefore analytic on the intersection of the domains of analyticity $\mathcal{B}(\mathcal{X}, \varrho_\varphi) \cap \mathcal{B}(\mathcal{X}, \varrho_g) = \mathcal{B}(\mathcal{X}, \varrho)$ with $\varrho = \min\{\varrho_g, \varrho_\varphi\}$.

It remains to prove that $e^{\langle \eta, \cdot \rangle} V_{t_u}(\cdot) \in L^1(\mathbb{R}^D)$. Here the Lipschitz continuity of f yields

$$||e^{\langle \eta, \cdot \rangle} V_{t_u}(\cdot)||_{L^1} \leq L_f \left(||e^{\langle \eta, \cdot \rangle} g_{t_u}(\cdot)||_{L^1} + ||e^{\langle \eta, \cdot \rangle} V_{t_{u+1}}(\cdot)||_{L^1} \right) < \infty. \qquad \square$$

Often, the discrete time problem (2.4) and (2.5) is an approximation of a continuous time problem, and thus we are interested in the error behavior for $n \to \infty$.

Remark 3.6. Assume the setup of Corollary 3.4. Moreover, assume that $\varepsilon_{tr} = \varepsilon_{gm} = 0$. If we let N and n go to infinity, we have to ensure that the error bound tends to zero. We use that $\varepsilon_{int}(\underline{\varrho}, N, D, \overline{B}) \leq C_1 \underline{\varrho}^{-\underline{N}}$ for a constant $C_1 > 0$ and $\underline{N} = \min_i N_i$. The following condition on the relation between n and N ensures convergence

$$n < \frac{\log(\varrho)}{C_1 D} \cdot \frac{\underline{N}}{\log(\Lambda_{\overline{N}}) + \log(L_f)} + 1.$$

4. Implementational aspects of the dynamic Chebyshev method. In this section, we discuss several approaches to compute the generalized moments (2.7) which contain the model dependent part. Moreover, preparing the numerical experiments, we tailor the dynamic Chebyshev method to the pricing of American put options.

4.1. Derivation of generalized moments. Naturally, the question arises how the generalized moments (2.7) can be derived. Here, we present four different ways and illustrate all approaches in the one-dimensional case $\mathcal{X} = [\underline{x}, \overline{x}]$. Similar formulas can be obtained for a multidimensional domain.

Probability density function. For the derivation of $\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k]$, let the density function of the random variable $X_{t_{u+1}}|X_{t_u} = x_k$ be given as $f^{u,k}(x)$. Then, the conditional expectation can be derived by solving an integral,

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] = \int_{\underline{x}}^{\overline{x}} T_j(\tau^{-1}_{[\underline{x},\overline{x}]}(y))\, f^{u,k}(y) \mathrm{d}y,$$

using $p_j(y) = T_j(\tau^{-1}_{\mathcal{X}}(y)) 1_{\mathcal{X}}(y)$. This approach is rather intuitive and easy to implement.

Fourier transformation. Assume the process X has stationary increments and the characteristic function φ of $X_{\Delta t}$ is explicitly available. We apply Parseval's identity (see [30]) and use Fourier transforms

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] = \int_{-\infty}^{\infty} p_j(x + x_k) F(\mathrm{d}x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} \widehat{p_j^{x_k}}(\xi) \varphi(-\xi) \mathrm{d}\xi,$$

where $p_j^{x_k}(x) = p_j(x + x_k)$. Using the definition of $\tau_{[\underline{x},\overline{x}]}$, we can express the Fourier transform of $p_j^{x_k}(x)$ with the help of the Chebyshev polynomial $T_j(y)$. This yields

$$\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] = \frac{1}{2\pi} e^{-i\xi x^k} e^{i\xi(\overline{x} - \frac{\overline{x}-\underline{x}}{2})} \frac{\overline{x} - \underline{x}}{2} \int_{-\infty}^{\infty} \widehat{T}_j\Big(\frac{\overline{x} - \underline{x}}{2}\xi\Big) \varphi(-\xi) \mathrm{d}\xi. \tag{4.1}$$

The Fourier transforms of the Chebyshev polynomials $\widehat{T_j}$ are presented in [7] and the authors also provide a MATLAB implementation.

Truncated moments. In this approach, we use that each one-dimensional Chebyshev polynomial can be represented as a sum of monomials, i.e.,

$$T_j(x) = \sum_{l=0}^{j} a_l x^l, \ j \in \mathbb{N}.$$

The coefficients a_l, $l = 0, \dots, j$, can easily be derived using the *chebfun* function *poly()*; see [8]. Then,

$$\begin{aligned}\mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] &= \mathbb{E}[T_j(\tau_{\mathcal{X}}^{-1}(X_{t_{u+1}}))1_{\mathcal{X}}(X_{t_{u+1}})|X_{t_u} = x_k] \\ &= \sum_{l=0}^{j} a_l \mathbb{E}[(\tau_{\mathcal{X}}^{-1}(X_{t_{u+1}}))^l 1_{\mathcal{X}}(X_{t_{u+1}})|X_{t_u} = x_k].\end{aligned}$$

As $\tau_{\mathcal{X}}$ is linear the computation of the generalized moments has thus been reduced to deriving truncated moments.

Monte Carlo simulation. Last, especially in cases for which neither a probability density function nor a characteristic function of the underlying process is given, Monte Carlo simulation is a suitable choice. For every nodal point x_k one simulates N_{MC} paths $X^i_{t_{u+1}}$ of $X_{t_{u+1}}$ with starting value $X_{t_u} = x_k$. These simulations can then be used to approximate

$$\Gamma_{t_u,t_{u+1}}(p_j)(x^k) = \mathbb{E}[p_j(X_{t_{u+1}})|X_{t_u} = x_k] \approx \frac{1}{N_{MC}} \sum_{i=1}^{N_{MC}} p_j(X^i_{t_{u+1}})$$

for every $j \in \mathcal{J}$. For an overview of Monte Carlo simulation from SDEs and variance reduction techniques we refer to [13] and [21].

4.2. Computational complexity of the algorithm. In this section, we investigate the complexity and thus the computational cost of the dynamic Chebyshev algorithm. In order to do so, the offline/online structure of the method has to be taken into account. We assume an equidistant time-stepping and that the stationarity assumption (2.7) holds.

In the offline step, we thus need to compute the $(N+1)^2$ generalized moments $\Gamma_{j,k} = \mathbb{E}[p_j(X_{\Delta t})|X_0 = x_k]$. When using numerical integration to compute the moments, the evaluation of the integrand at M_{quad} quadrature points is required and similarly for the Monte Carlo approach on M_{MC} samples. In total, the complexity of the offline phase scales with $N^2 M_{quad}$ or $N^2 M_{MC}$. The complexity can be reduced when the straightforward approach for the moment calculation is replaced with a more sophisticated approach. Moreover, parallelization can help to reduce the runtime significantly. It is important to acknowledge that the three quantities N, M_{quad}, and M_{MC} are on a different scale. The number of Monte Carlo simulations is typically much higher than the number of quadrature points or Chebyshev nodes. For example, $50,000$ might be a good choice for M_{MC} whereas for $M_{quad} = 500$ might already be high enough. Later on, we will investigate the optimal relation between N and M_{MC} in more detail.

After the offline phase all model-dependent quantities are readily available. Moreover, the effort of the offline phase is independent of the number of time steps n or the number of different payoffs that have to be priced.

In the online phase, we need to compute the vector of nodal values $\widehat{V}_{t_u}(x_k)$ for all $k = 0, \dots, N$ and then the coefficient vector $c_j^{t_u}$ for $j = 0, \dots, N$ in every time step.

Both require the multiplication of a vector with $N+1$ entries with a $(N+1)\times(N+1)$ matrix. Hence, the total effort scales with nN^2, where n is the number of time steps. The complexity of the online phase is therefore independent of the numerical method applied in the offline calculations. Since N is typically relatively small the method becomes very efficient.

4.3. Application to option pricing. In the numerical section we use the dynamic Chebyshev method to price a barrier option and an American put option. Assuming an asset model of the form $S_t = e^{X_t}$, the DPP for an American put option becomes

$$\begin{aligned} V_T(x) &= (K - e^x)^+, \\ V_{t_u}(x) &= \max\left\{(K-e^x)^+, e^{-r(t_{u+1}-t_u)}\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u}=x]\right\}. \end{aligned}$$

The general formulation of the DPP (2.4) and (2.5) includes also the pricing of different types of discretely monitored barrier options. One example is an up-and-out call option with value function

$$\begin{aligned} V_T(x) &= (e^x - K)^+ \mathbb{1}_{(-\infty,b]}(x), \\ V_{t_u}(x) &= e^{-r(t_{u+1}-t_u)}\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u}=x]\mathbb{1}_{(-\infty,b]}(x) \end{aligned}$$

with strike K and barrier b. Other examples are a down-and-out put option or options with multiple barriers.

Reducing the truncation error. Typically, the support of the underlying process X_t is $\mathbb{R}$ and the restriction to a compact domain $\mathcal{X} = [\underline{x}, \overline{x}]$ introduces a truncation error. The error is less relevant for barrier options than for American options since the option value is zero for $x > b$ and thus the truncation error for large x is zero if $\overline{x} = b$. For small x, we know that the option value converges toward zero and the truncation error becomes small for $\overline{x}$ small enough. For an American put option we exploit the asymptotic behavior of the payoff in order to reduce the truncation error. If X_{t_u} is below the exercise boundary, the option is exercised at the value $K - e^{X_{t_u}}$, which we exploit for $X_{t_u} < \underline{x}$. The function $x \mapsto V_{t_u}$ tends to zero from above for $x \to \infty$ and thus for $\overline{x}$ large enough the truncation to zero for $x > \overline{x}$ is justified. Hence, we introduce the following modification of the dynamic Chebyshev method,

$$\begin{aligned} V_{t_{u+1}}(x) &= V_{t_{u+1}}(x)1_{\{x<\underline{x}\}} + V_{t_{u+1}}(x)1_{\{x\in\mathcal{X}\}} + V_{t_{u+1}}(x)1_{\{x>\overline{x}\}}, \\ &\approx (K - e^x)1_{\{x<\underline{x}\}} + \widehat{V}_{t_{u+1}}(x)1_{\{x\in\mathcal{X}\}}, \end{aligned}$$

and thus

$$\begin{aligned} \mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})|X_{t_u}=x_k] \approx{}& \mathbb{E}[(K - e^{X_{t_{u+1}}})1_{\{X_{t_{u+1}}<\underline{x}\}}|X_{t_u}=x_k] \\ &+ \sum_{j=0}^{N} c_j(t_{u+1})\Gamma_{j,k}(t_u) \end{aligned}$$

for $\underline{x}$ small and $\overline{x}$ large enough. One can precompute $\mathbb{E}[(K-e^{X_{t_{u+1}}})1_{\{X_{t_{u+1}}<\underline{x}\}}|X_{t_u} = x_k]$. We emphasize that similar modifications to reduce the truncation error can be found for other payoff profiles, e.g., for digitals, butterfly options, or any other combination of different put options.

Smoothing initial step. Moreover, we also modify the first time step from t_n to t_{n-1}. At time t_{n-1} we need to calculate the values at the nodal points

$$V_{t_{n-1}}(x_k) = f(g(x_k), \mathbb{E}[V_T(X_T)|X_{t_{n-1}} = x_k]).$$

Instead of approximating V_T with Chebyshev polynomials and using (2.6) we can directly exploit that $V_T(x) = g(x)$ and calculate

$$V_{t_{n-1}}(x_k) = f(g(x_k), \mathbb{E}[g(X_T)|X_{t_{n-1}} = x_k]). \tag{4.2}$$

This means we need to calculate the conditional expectations $\mathbb{E}[g(X_T)|X_{t_{n-1}} = x_k]$ for $k = 0, \dots, N$, which are essentially European option prices. From [11], we know that European option prices are analytic functions of the starting value x for a large class stock price models and payoffs. Hence, the kink of the payoff is no longer relevant and the method is "smoothed," i.e., convergence is improved. In section 5.3, we will investigate the effect of the smoothing on the error decay numerically.

Note that this implementation of the initial step introduces a smoothing effect but no additional error; on the contrary, by computing the generalized moments (4.2) directly we avoid a polyonmial approximation of the known payoff function. Hence the modification improves the accuracy of the method.

The option's sensitivities. The option's sensitivities Delta and Gamma can be computed by taking the first or second derivative of

$$S \mapsto \widehat{V}_0(\log(S)) = \sum_{j \in \mathcal{J}} c_j(t_0) p_j(\log(S)).$$

Thus Delta and Gamma are expressed as the sum of derivatives of Chebyshev polynomials. In particular, their derivation comes without any additional computational costs in the offline phase or in the time-stepping.

5. Numerical experiments. In this section, we use the dynamic Chebyshev method to price American put options and we numerically investigate the convergence of the method. Moreover, we compare the method with the least-squares Monte Carlo method of [23]. All experiments were performed on a computer with Intel Core i7-6700 with 3.4GHz and 16GB memory. All codes are written in MATLAB version R2017b.

5.1. Stock price models. For the convergence analysis we use three different stock price models.

The Black–Scholes model. In the classical model of [3] the stock price process is modelled by the SDE

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma S_t \mathrm{d}W_t,$$

where r is the risk-free interest rate and $\sigma > 0$ is the volatility. In this model the log-returns $X_t = \log(S_t)$ are normally distributed and for the double truncated moments

$$\mathbb{E}[X^m 1_{[a,b]}(X)] \qquad \text{for} \qquad X \sim \mathcal{N}(\mu, \sigma^2)$$

explicit formulas are available. Kan and Robotti [20] present results for the (multivariate) truncated moments and provide a MATLAB implementation.

The Merton jump diffusion model. The jump diffusion model introduced by [26] adds jumps to the classical Black–Scholes model. For $S_t = S_0 e^{X_t}$ the log-returns X_t follow a jump diffusion with volatility σ and added jumps arriving at rate $\lambda > 0$ with normal distributed jump sizes according to $\mathcal{N}(\alpha, \beta^2)$. From [6] we obtain the characteristic function of X_t given by

$$\varphi(z) = exp\left(t\left(ibz - \frac{\sigma^2}{2}z^2 + \lambda\left(e^{iz\alpha - \frac{\beta^2}{2}z^2} - 1\right)\right)\right)$$

with risk-neutral drift $b = r - \frac{\sigma^2}{2} - \lambda(e^{\alpha + \frac{\beta^2}{2}} - 1)$.

The constant elasticity of variance model. CEV model as stated in [32] is a local volatility model based on the stochastic process

$$\mathrm{d}S_t = rS_t\mathrm{d}t + \sigma S_t^{\beta/2}\mathrm{d}W_t \quad \text{for} \quad \beta > 0. \tag{5.1}$$

Hence the stock volatility $\sigma S_t^{(\beta-2)/2}$ depends on the current level of the stock price. For the special case $\beta = 2$ the model coincides with the Black–Scholes model. However, from market data one typically observes a $\beta < 2$. The CEV-model is one example of a model which has neither a probability density, nor a characteristic function in closed-form.

5.2. Expected convergence behavior. Before we perform a numerical convergence analysis, we recall the theoretical error analysis and point out what type of error decay we can expect.

First, we consider an analytic value function and omit a possible truncation error. In this case we know from Corollary 3.4 that the following error bound holds,

$$\varepsilon_{t_u} \leq c_1 \varrho^{-N} \log(N)^{Dn} + c_2 \log(N)^{Dn}\overline{\delta},$$

or with $\delta^\star$ instead of $\overline{\delta}$ if (GM*) holds. This yields for the log-error

$$\begin{aligned} \log(\varepsilon_{t_u}) &\leq \log\left(c_1 \varrho^{-N} \log(N)^{Dn} + c_2 \log(N)^{Dn}\overline{\delta}\right) \\ &= \log\left(c_1 \varrho^{-N} \log(N)^{Dn}\right) + \log\left(1 + \frac{c_2}{c_1}\varrho^N\overline{\delta}\right) \\ &= \log(c_1) - \log(\varrho)N + Dn\log(\log(N)) + \log\left(1 + \frac{c_2}{c_1}\varrho^N\overline{\delta}\right). \end{aligned} \tag{5.2}$$

If we assume that $\varrho^N\overline{\delta} \leq 1$ the log-error as a function of N should be bounded by a function of the form $N \mapsto c - m_1 N + m_2 \log(\log(N))$ for constants $c, m_1, m_2 > 0$ and since the linear term dominates the log log-term, we expect to observe an exponential error decay in N. For $\varrho^N\overline{\delta} > 1$ we obtain

$$\begin{aligned} \log\left(1 + \frac{c_2}{c_1}\varrho^N\overline{\delta}\right) &= \log\left(\frac{c_2}{c_1}\varrho^N\overline{\delta}\right) + \log\left(\frac{c_1}{c_2}\varrho^{-N}\overline{\delta}^{-1} + 1\right) \\ &\leq \log\left(\frac{c_2}{c_1}\right) + \log(\varrho)N + \log(\overline{\delta}) + \log\left(\frac{c_1}{c_2} + 1\right). \end{aligned}$$

When we plug this term into (5.2), the terms $-\log(\varrho)N$ and $\log(\varrho)N$ cancel each other out. Combining the two cases yields two observations for the convergence behavior. If the error when computing the generalized moments in the offline phase is $\overline{\delta}$ or $\delta^\star$ then the total error of the method will decrease in N until the level $\overline{\delta}$, respectively

$\delta^\star$, is reached. Second, in terms of the total computational effort one should choose N subject to the accuracy of the generalized moments. For example, in the Monte Carlo case the optimal N is a function of the number of simulations or of the number of quadrature points in the Fourier case. In the Monte Carlo case the error $\delta^\star$ decays typically with $cM^{-0.5}$ and from $\varrho^N\delta^\star \leq 1$ follows $N \leq \tilde{c}\log(M)$ for some constant $\tilde{c} > 0$. If we fix N like this, the complexity of the offline phase N^2M becomes $\log^2(M)M$ and the complexity of the online phase N^2 becomes $\log^2(M)$. In the Fourier case the error $\overline{\delta}$ depends on the regularity of the integrand which is model dependent. Typically the error will decrease much faster than the Monte Carlo error.

If the value function is only continuously differentiable and not analytic, we can no longer apply the results from section 3. Nevertheless, we can use the dynamic Chebyshev method and come up with a very rough estimate of the expected convergence behavior. We know that the convergence of the Chebyshev interpolation is of polynomial order for continuously differentiable functions. If we simply replace the term ϱ^{-N} by a term N^{-p} for a $p \in \mathbb{N}$, we can perform the same calculations as in the anlytic case. We obtain for the log-error

$$\log(\varepsilon_{t_u}) \leq \log(c_1) - p\log(N) + Dn\log(\log(N)) + \log\left(1 + \frac{c_1}{c_2}N^{-p}\overline{\delta}\right).$$

Assuming $N^p\overline{\delta} \leq 1$ suggests that the log-error as a function of N is bounded by a function $N \mapsto c - m_1\log(N) + m_2\log(\log(N))$ for constants $c, m_1, m_2 > 0$. In this case the log-error is approximately linear in $\log(N)$. If we use Monte Carlo simulation in the offline step with decay $cM^{-0.5}$, the condition $N^p\delta^\star \leq 1$ implies for $p = 1$ that $N \leq \tilde{c}\sqrt{M}$ and more generally $N \leq \tilde{c}M^{0.5/p}$. Similarly to the analytic case, if we choose $N = \tilde{c}M^{0.5/p}$, the complexity of the offline phase N^2M becomes $M^{1+1/p}$ and the complexity of the online phase nN^2 becomes $nM^{1/p}$.

5.3. Numerical convergence analysis. In this section we investigate the convergence of the dynamic Chebyshev method. We price a barrier call option and an American put option along with the options' Delta and Gamma in the Black–Scholes and the Merton jump diffusion model, where we can use the COS method of [10] as a benchmark. The COS method is based on the Fourier-cosine expansion of the density function and provides fast and accurate results for the class of Lévy models. We use the implementations of method which were provided for the benchmarking project of [34]. The provided implementations are slightly modified to fit for our examples.

For the experiments, we use the following parameter sets in the Black–Scholes model

$$K = 100, \quad r = 0.03, \quad \sigma = 0.25, \quad T = 1,$$

and for the Merton jump diffusion model

$$K = 100, \quad r = 0.03, \quad \alpha = -0.5, \quad \beta = 0.4, \quad \sigma = 0.25, \quad \lambda = 0.4,$$

and we use 32 time steps. The jump parameters α, β, and λ are taken from [34].

5.3.1. Convergence for analytic value functions. We price a barrier option with call payoff $g_T(x) = (e^x - K)^+\mathbb{1}_{(\infty,b]}(x)$ and barrier $b = \log(125)$ in the Black–Scholes model. Defining $\mathcal{X} = [\underline{x}, b]$, $g_{t_u}(x) = \mathbb{1}_{(\infty,b]}(x)$ and $f(x,y) = xy$ the pricing problem of a barrier option as introduced in section 4.3 can be written in the general form of (2.4) and (2.5).

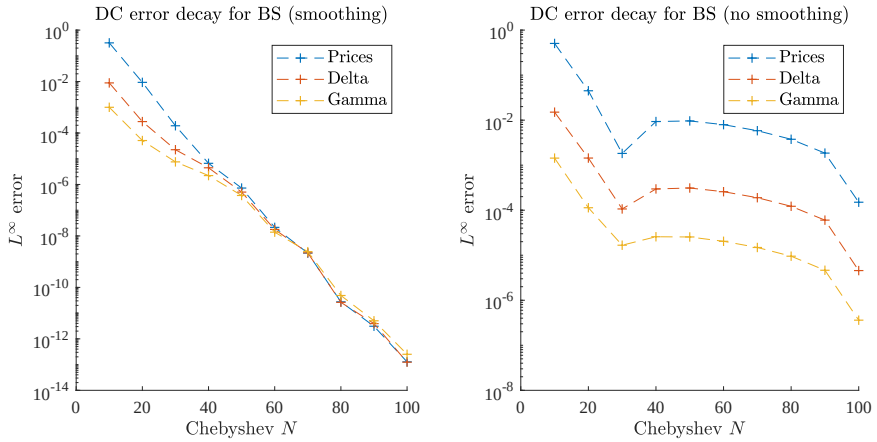

FIG. 5.1. *Error decay prices of dynamic Chebyshev in the Black–Scholes model using smoothing in the first time step (left) and without smoothing (right). The conditional expectation of the Chebyshev polynomials are calculated with the density function.*

Remark 5.1. When conditions (i) and (ii) of Proposition 3.5 hold and the smoothing of the payoff as stated in (4.2) is applied then the value function of an up-and-out barrier option

$$V_{t_u} : [\underline{x}, b] \ni x \mapsto \mathbb{E}[V_{t_u}(X_{t_{u+1}})|X_{t_u} = x]\mathbb{1}_{(-\infty,b]}(x)$$

is a function with an analytic extension to some Bernstein ellipse $\mathcal{B}([\underline{x}, b], \varrho)$, $\varrho \in (1, \infty)$ for all $u = 0, \ldots, n-1$. In this case the convergence result for the dynamic Chebyshev method of Corollary 3.4 holds.

From [11] we know that conditions (i) and (ii) of Proposition 3.5 are fulfilled in the Black–Scholes model and thus we can expect the log-error to decay approximately linearly in the number of Chebyshev nodes.

For the following experiments, we used the density approach to calculate the generalized moments implemented with the MATLAB quadrature routine *quadgk* with an absolute as well as relative error tolerance of 10^{-13} and we set $\underline{x} = \log(10)$. Prices and sensitivities are calculated on a grid of starting values equally distributed between 90 and 110 and compared to the benchmark method. The left plot in Figure 5.1 shows the log-error for an increasing number of nodes $N = 10, \ldots, 100$. The log-error for the prices as well as for Delta and Gamma decays linearly in N as we expected and reaches an accuracy below 10^{-12}. With only 50 nodal points the method is already able to achieve an accuracy below 10^{-6}.

The right plot in Figure 5.1 shows the same experiment without the smoothing in the initial time step. The method still converges but the decay of the log-error is no longer linear.

5.3.2. Convergence for differentiable value functions. Next, we price an American put option in the Black–Scholes and in the Merton model. Here, the value function is only continuously differentiable but not analytic. Due to the maxima function in the evaluation of the value function in every time step, we cannot expect that the function is analytic around the optimal exercise point. However, Peskir and Shiryaev [28] show that the value function of an American option is still continuously differentiable at the exercise point. This property is often called the "smooth-fit"

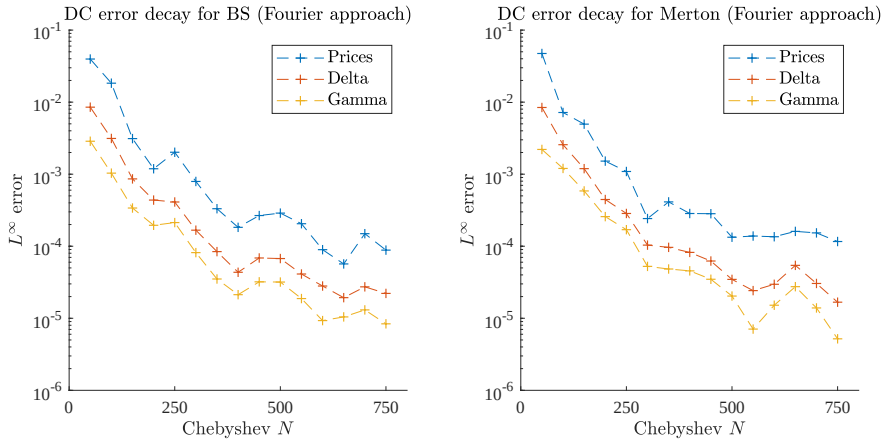

FIG. 5.2. *Error decay prices of dynamic Chebyshev in the Black–Scholes model (left) and the Merton model (right). The conditional expectation of the Chebyshev polynomials are calculated with the Fourier transformation.*

property. Bayraktar [1] shows that this smoothness property holds also in a jump-diffusion model. The expected decay of the log-error is therefore slower than a linear decay and behaves approximately like $-p\log(N)$. Therefore, we should need more nodal points as in the analytic case to obtain the same accuracy.

For both models the generalized moments are computed by the Fourier approach as stated in (4.1). We truncate the integral at $|\xi| \leq 250$ and use Clenshaw–Curtis with 500 nodes for the numerical integration. For the Fourier transform of the Chebyshev polynomials the implementation of [7] is used. We run the dynamic Chebyshev method for an increasing number of Chebyshev nodes $N = 50, 100, \ldots, 750$. Then, option prices and their sensitivities delta and gamma are calculated on a grid of different values of S_0 equally distributed between 60% and 140% of the strike K. The resulting prices and Greeks are compared using the COS method as a benchmark and the maximum error over the grid is calculated. Here we use the implementation provided in [34].

Figure 5.2 shows the error decay for the Black–Scholes model (left-hand side) and the Merton model (right-hand side). We observe that the method converges and an error below 10^{-3} is reached for $N = 300$ Chebyshev nodes. The speed of the convergence is similar for both stock price models and slower than in the barrier option example. The plots demonstrate an approximately polynomial error decay in N. Hence, the experiments confirm that the method can be used for an American put option.

5.3.3. Convergence for a bivariate barrier option. In this section, we provide evidence that the method also works for multivariate options by looking at the convergence of the dynamic Chebyshev method for barrier options with two barriers. We consider two assets S_t^1, S_t^2 and an option with payoff

$$g(x_1, x_2) = (e^{x_1} - K)^+ 1_{(-\infty, b_1]}(x_1) 1\!\!1_{(-\infty, b_2]}(x_2) \quad \text{with} \quad x_1 = \log(S_1),\ x_2 = \log(S_2)$$

strike K and barrier b. This type of option is also referred to as an outside/rainbow barrier option as a second (outside) underlying asset is included to generate an additional discount compared to a standard (barrier) option. An economic example could

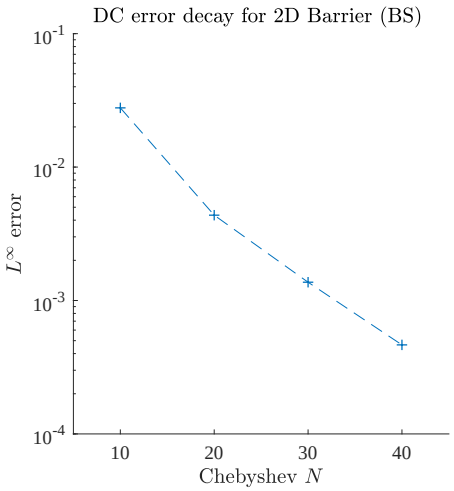

FIG. 5.3. *Error decay of the dynamic Chebyshev approach for a bivariate barrier option in a multivariate Black–Scholes model. The conditional expectation of the Chebyshev polynomials are calculated using the density function.*

be that a company would like to hedge against increasing prices of a commodity only if the economy stays at or below its current level. In the scenario of an economic boom higher prices cover the increase in costs and no hedge is required. Different examples of options with multiple barriers and their economic interpretation can, for example, be found in [9].

Here, we assume that both assets follow a geometric Brownian motion and hence we are in a bivariate Black–Scholes type model. We fix the following model parameters,

$$K = 100, \quad r = 0.03, \quad \sigma_1 = 0.25, \quad \sigma_2 = 0.2, \quad \rho = 0.4, \quad T = 1,$$

and choose as a barrier $b_1 = \log(125)$ and $b_2 = \log(120)$. For the calculation of the generalized moments we use the density approach implemented using the MATLAB function *integral*2 with an absolute and relative error tolerance of 10^{-6}. We run the dynamic Chebyshev method for an increasing number of points $N = N_1 = N_2$ ranging from 10 to 40 and calculate prices on a two-dimensional grid of starting values equally distributed in $[90, 110] \times [90, 110]$. For the comparison of prices, we run the method with $N = 50$.

Figure 5.3 shows the resulting error decay. We still observe that the log-error decays almost linearly, in N, where N corresponds to a total number of N^2 grid points. In a general D-dimensional framework we can expect to need N^D points for the same error behavior in N. This is often called the course of dimensionality. Different numerical techniques have been developed to tackle this problem such as low-rank tensor techniques and sparse grids. These can be exploited when applying the dynamic Chebyshev method to multivariate pricing problems.

5.4. Dynamic Chebyshev with Monte Carlo. So far, we have empirically investigated the error decay of the method for option prices and their derivatives. In this section, we compare the dynamic Chebyshev method with the least-squares Monte Carlo approach of [23] in terms of accuracy and runtime.

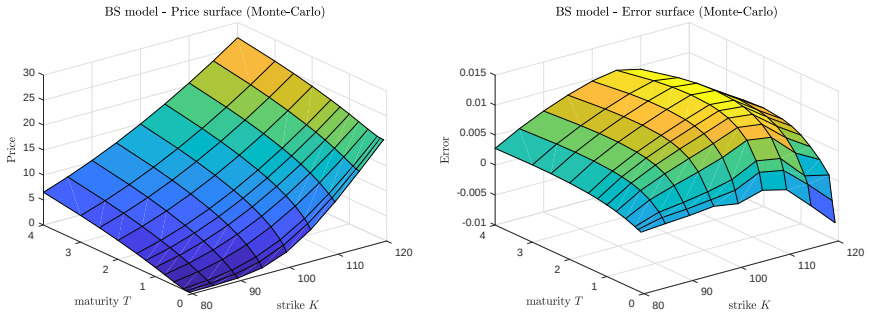

FIG. 5.4. *Price surface and corresponding error of the dynamic Chebyshev method in the Black–Scholes model. The conditional expectations are calculated with Monte Carlo.*

5.4.1. The Black–Scholes model. As a first benchmark, we use the Black–Scholes model with an interest rate of $r = 0.03$ and volatility $\sigma = 0.25$. Here, we look at a whole option price surface with varying maturities and strikes. We choose 12 different maturities between one month and four years given by

$$T \in \{1/12, 2/12, 3/12, 6/12, 9/12, 1, 15/12, 18/12, 2, 30/12, 3, 4\}$$

and strikes equally distributed between 80% and 120% of the current stock price $S_0 = 100$ in steps of 5%. We fix $n = 504$ time steps (i.e., exercise rights) per year, which is equivalent to two ticks per trading day (assuming 252 trading days per year). We use a relatively high number of exercise rights to ensure that the solution in discrete time is a good approximation of the actually continuous time problem of pricing an American put.

We compare the dynamic Chebyshev method to the least-squares Monte Carlo approach. We run both methods for an increasing number of Monte Carlo paths according to

$$M \in \{2500, 5000, 10000, 20000, 40000, 80000\}\,. \tag{5.3}$$

The convergence of the dynamic Chebyshev method depends on both the number of nodes N and the number of Monte Carlo paths M. For an optimal convergence behavior one needs to find a reasonable relationship between these factors. The analysis of the expected convergence behavior in section 5.2 shows that the number of Chebyshev nodes N should be $c\sqrt{M}$ for a constant $c > 0$. Numerical experiments indicated that $c = \sqrt{2}$ is a very reasonable choice and thus we fix $N = \sqrt{2}\sqrt{M}$.

Figure 5.4 shows the price surface and the error surface for $N = 400$ and $M = 80000$. The error was estimated by using the COS method as benchmark. We reach a maximal error below 0.015 on the whole option surface.

In Figure 5.5 the log_{10}-error is shown as a function of the log_{10}-runtime for both methods. The left figure compares the total runtimes and the right figure compares the offline runtime. For the dynamic Chebyshev method the total runtime includes the offline-phase and the online phase. The offline-phase consists of the simulation of one time step of the underlying asset process $X_{\Delta t}$ for $N + 1$ starting values $X_0 = x_k$ and of the computation of the conditional expectations $E[p_j(X_{\Delta t})|X_0 = x_k]$ for $j, k = 0, \ldots, N$. The online phase is the actual pricing of the American option for all strikes and maturities. Similarly, the total runtime of the least-squares Monte Carlo method includes the simulation of the Monte Carlo paths (offline-phase) and the pricing of the option via backward induction (online-phase).

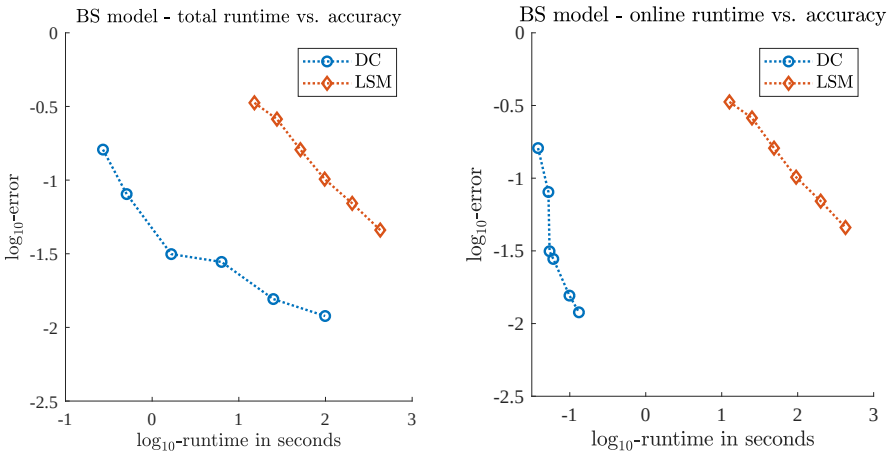

FIG. 5.5. *Log-Log plot of the total/online runtime vs. accuracy. Comparison of the dynamic Chebyshev method with the least-squares Monte Carlo algorithm.*

We observe that the dynamic Chebyshev method reaches the same accuracy with a much lower runtime. For example, a maximum error of 0.1 is reached in a total runtime of $0.5s$ with the dynamic Chebyshev method whereas the LSM approach needs $98s$. This means the dynamic Chebyshev method is nearly 200 times faster for the same accuracy. For the actual pricing in the online phase, the gain in efficiency is even higher. We observe that the dynamic Chebyshev method outperforms the least-squares Monte Carlo method in terms of the total runtime and the pure online runtime. Moreover, we observe that the performance gain from splitting the computation into an offline and an online phase is much higher for the dynamic Chebyshev method. For instance, in the example above the online runtime of the dynamic Chebyshev method is $0.05s$ whereas the LSM takes $95s$, a factor of 1900 times more.

The main advantage of the dynamic Chebyshev method is that once the conditional expectations are calculated, they can be used to price the whole option surface. The pure pricing, i.e., the online phase, is highly efficient. Furthermore, one only needs to simulate one time step Δt of the underlying stochastic process instead of the complete path. We investigate this efficiency gain by varying the number of options and the number of time-steps (exercise rights). From section 4.2, we know that the computational complexity of the offline phase is independent of the number of time-steps and the number of payoffs/options we want to price. Once the generalized moments are calculated the pricing of an option requires only one run of online time stepping. Figure 5.5 shows that the online runtime even for pricing a complete option surface is less than 1% of the total runtime. Therefore we can expect that varying the number of options and the number of exercise rights has nearly no effect on the total runtime of the dynamic Chebyshev method.

Figure 5.6 compares the total runtime of the dynamic Chebyshev method with the total runtime of the LSM method for an increasing number of options and for an increasing number of time steps. As expected, we can empirically confirm that the efficiency gain by the dynamic Chebyshev methods increases with the number of options and the number of exercise rights. In both cases, the runtime of the dynamic Chebyshev method stays nearly constant whereas the runtime of the LSM method increases approximately linearly.

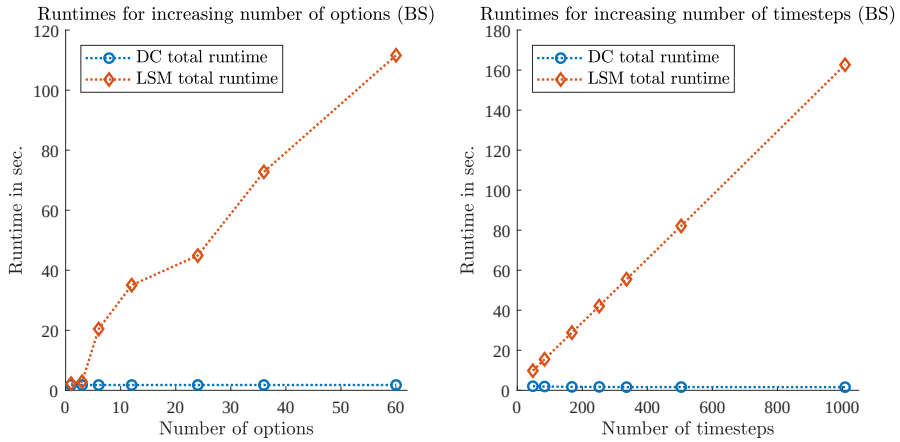

FIG. 5.6. *Total runtime of the dynamic Chebyshev and the LSM method for an increasing number of options (left) and an increasing number of timesteps (right).*

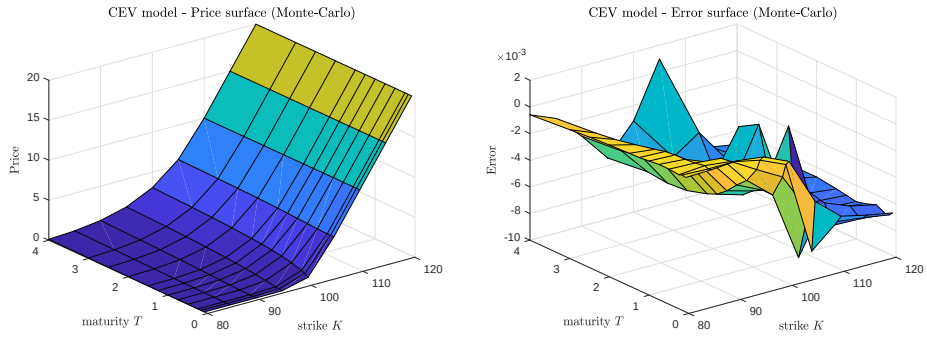

FIG. 5.7. *Price surface and corresponding error of the dynamic Chebyshev method in the CEV model. The conditional expectations are calculated with Monte Carlo.*

5.4.2. The CEV model. Next, we use the CEV model for the underlying stock price process. We perform the same experiments as in the last section. The parameters in the CEV model, as in (5.1), are the following:

$$\sigma = 0.25, \quad r = 0.03, \quad \beta = 1.5.$$

Similarly, we compare the dynamic Chebyshev and the LSM method by computing the prices of an option price surface. We use the same parameter specifications for K, T and n. We run both methods for an increasing number of Monte Carlo simulations M and fix $N = \sqrt{2}\sqrt{M}$. Figure 5.7 shows the price surface and the error surface for $N = 400$ and $M = 80000$. The error is calculated using a binomial tree implementation of the CEV model based on [27].

In Figure 5.8 the log_{10}-error is shown as a function of the log_{10}-runtime for both methods. The left figure compares the total runtimes and the right figure compares the offline runtimes. Again, we observe that the dynamic Chebyshev method is faster for the same accuracy and it profits more from an offline-online decomposition. For example, the total runtime of the dynamic Chebyshev method to reach an accuracy below 0.03 is $3.5s$ whereas LSM takes $136s$. For the online runtimes this out-performance is $1s$ to $122s$.

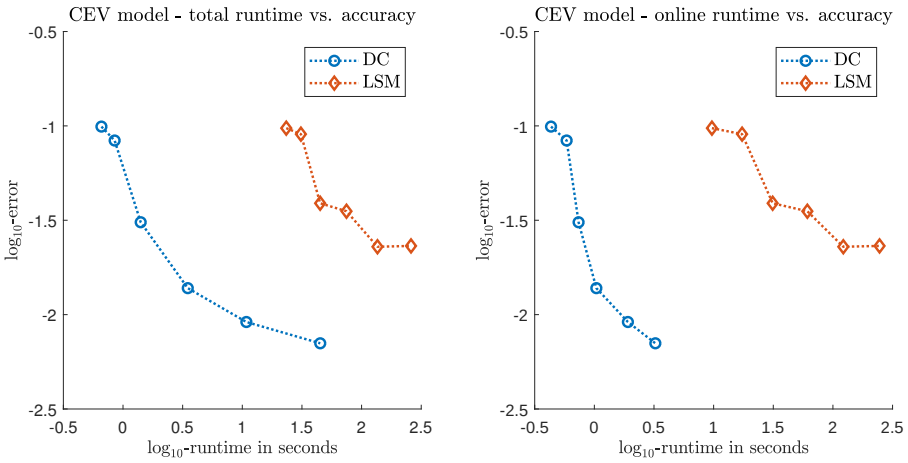

FIG. 5.8. *Log-Log plot of the total/online runtime vs. accuracy. Comparison of the dynamic Chebyshev method with the least-squares Monte Carlo algorithm.*

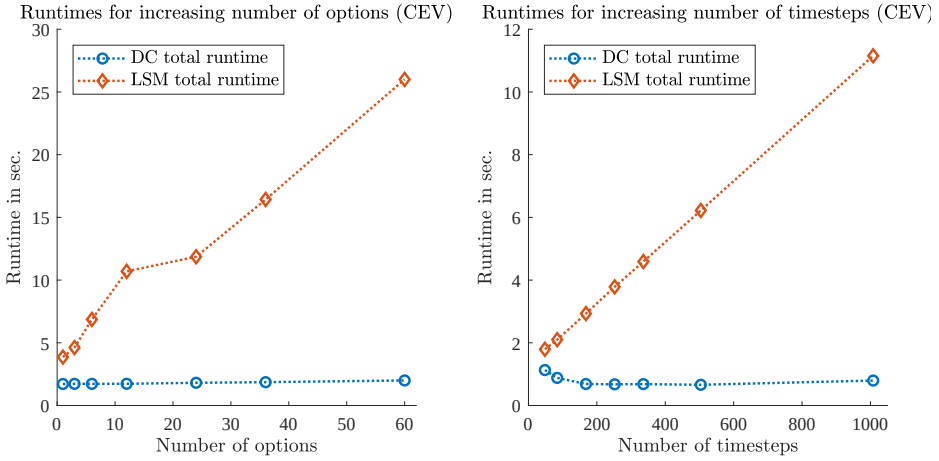

FIG. 5.9. *Total runtime of the dynamic Chebyshev and the LSM method for an increasing number of options (left) and an increasing number of timesteps (right).*

Investigating this efficiency gain further, we look at the performance for different numbers of options and time steps (exercise rights). Similarly to the last section, Figure 5.9 compares the total runtime of the dynamic Chebyshev method with the total runtime of the LSM method for an increasing number of options and time steps. In both cases, the runtime of the dynamic Chebyshev method stays nearly constant whereas the runtime of the LSM method increases approximately linearly. This observation is consistent with the theoretical considerations in section 4.2 and the findings for the Black–Scholes model in the previous section.

6. Conclusion and outlook. We have introduced a new approach to price American options via backward induction by approximating the value function with Chebyshev polynomials. Thereby, the computation of the conditional expectation of the value function in each time step is reduced to the computation of conditional expectations of polynomials. The proposed method separates the pricing of an option

into a part which is model-dependent (the computation of the conditional expectations) and the pure pricing of a given payoff which becomes independent of the underlying model. The first step, the computation of the conditional expectation of the Chebyshev polynomials, is the so-called offline phase of the method. The design of the method admits several qualitative advantageous:

- If the conditional expectations are set up once, we can use them for the pricing of many different options. Thus, the actual pricing in the online step becomes very simple and fast.
- In the precomputation step one can combine the method with different techniques, such as Fourier approaches and Monte Carlo simulation. Hence the method can be applied in a variety of models.
- The proposed approach is very general and flexible and thus not restricted to the pricing of American options. It can be used to solve a large class of optimal stopping problems.
- We obtain a closed-form approximation of the option price as a function of the stock price at every time step. This approximation can be used to compute the option's sensitivities Delta and Gamma at no additional costs. This holds for all models and payoff profiles, even if Monte Carlo is required in the offline phase.
- The method is easy to implement and to maintain. The precomputation step is well-suited for parallelization to speed up the method.

We have investigated the theoretical error behavior of the method and introduced explicit error bounds. We put particular emphasis on the combination of the method with Monte Carlo simulation. Numerical experiments confirm that the method performs well for the pricing of American options. A detailed comparison of the method with the least-squares Monte Carlo approach proposed by [23] confirmed a high efficiency gain, especially, when a high number of options is priced, for example, a whole option price surface. In this case, the dynamic Chebyshev method highly profits from the offline-online decomposition. Once the conditional expectations are computed, they can be used to price options with different maturities and strikes. Besides the efficiency gain, the closed-form approximation of the price function is a significant advantage because it allows us to calculate the sensitivities. Since [23] introduced their method different modifications have been introduced, either to increase efficiency or to tackle the sensitivities. For example, the simulation algorithm of [18] is comparable to LSM in terms of efficiency but is able to compute the Greeks at no additional costs. Moreover dual approaches were developed to obtain upper bounds for the option price; see [29] and more recently [2]. The presented experiments focused on the one-dimensional case with a bivariate application. A thorough numerical investigation of the multivariate case including a performance study will be presented in a follow-up paper.

The presented error analysis of the method under an analyticity assumption is the starting point for further theoretical investigations in the case of piecewise analyticity and (piecewise) differentiability. The former allows one to cover rigorously the American option pricing problem, and a preliminary version is presented in [25]. The qualitative merits of the method can be exploited in a variety of applications. Glau, Pachon, and Pötz [14] take advantage of the closed-form approximation to efficiently compute the expected exposure of early-exercise options as a step in CVA calculation. Moreover, the method can be used to price different options such as other types of barrier options, swing options, or multivariate American options.

Appendix A. Proof of Theorem 3.3.

Proof. Consider a DPP as defined in Definition 2.1, i.e., we have a Lipschitz continuous function

$$|f(x_1, y_1) - f(x_2, y_2)| \leq L_f(|x_1 - x_2| + |y_1 - y_2|).$$

Assume that the regularity Assumption 3.2 and the assumption on the truncation error (TR) hold. Then we have to distinguish between the deterministic case (GM) and the stochastic case (GM*). In the first case, the expectation in the error bound can simply be ignored. First, we apply Proposition 3.1. At time point T there is no random part and no distortion error. Thus,

$$\max_{x\in\mathcal{X}} \mathbb{E}\left[|V_T(x) - \widehat{V}_T(x)|\right] = \max_{x\in\mathcal{X}} |V_T(x) - \widehat{V}_T(x)| \leq \varepsilon_{int}(\varrho_{t_n}, N, D, M_{t_n}).$$

For ease of notation we will from now on write $\varepsilon^j_{int} = \varepsilon_{int}(\varrho_{t_j}, N, D, M_{t_j})$. We obtain for the error at t_u

$$\max_{x\in\mathcal{X}} \mathbb{E}\left[|V_{t_u}(x) - \widehat{V}_{t_u}(x)|\right] \leq \varepsilon^u_{int} + \Lambda_{\overline{N}} F(f, t_u) \tag{A.1}$$

with maximal distortion error $F(f, t_u) = \max_{k\in\mathcal{J}} \mathbb{E}\left[|V_{t_u}(x^k) - \widehat{V}_{t_u}(x^k)|\right]$.

Note that whether (GM) or (GM*) hold an approximation error of the conditional expectation of $\widehat{V}_{t_{u+1}}$ is made, i.e., $\mathbb{E}[\widehat{V}_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x^k] = \Gamma^k_{t_u}(\widehat{V}_{t_{u+1}}) \approx \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})$. The Lipschitz continuity of f yields

$$\begin{aligned}
\left|V_{t_u}(x^k) - \widehat{V}_{t_u}(x^k)\right| &= \left|f\left(g(t_u, x^k), \Gamma^k_{t_u}(V_{t_{u+1}})\right) - f\Big(g(t_u, x^k), \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\Big)\right| \\
&\leq L_f\Big(\left|g(t_u, x^k) - g(t_u, x^k)\right| + \left|\Gamma^k_{t_u}(V_{t_{u+1}}) - \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\right|\Big) \\
&= L_f\Big(\left|\Gamma^k_{t_u}(V_{t_{u+1}}) - \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\right|\Big) \\
&\leq L_f\Big(\left|\Gamma^k_{t_u}(V_{t_{u+1}}1_{\mathcal{X}}) - \Gamma^k_{t_u}(\widehat{V}_{t_{u+1}})\right| + \left|\Gamma^k_{t_u}(V_{t_{u+1}}1_{\mathbb{R}^D\setminus\mathcal{X}})\right| \\
&\qquad + \left|\Gamma^k_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\right|\Big).
\end{aligned}$$

Next, we consider the expectation for each of the three error terms. For the first term we obtain

$$\begin{aligned}
\mathbb{E}\left[\left|\Gamma^k_{t_u}(V_{t_{u+1}}1_{\mathcal{X}}) - \Gamma^k_{t_u}(\widehat{V}_{t_{u+1}})\right|\right] &= \mathbb{E}\left[\left|\mathbb{E}[V_{t_{u+1}}(X_{t_{u+1}})\mathbb{1}_{\mathcal{X}} - \widehat{V}_{t_{u+1}}(X_{t_{u+1}})|X_{t_u} = x^k]\right|\right] \\
&\leq \max_{x\in\mathcal{X}} \mathbb{E}\left[|V_{t_{u+1}}(x) - \widehat{V}_{t_{u+1}}(x)|\right] = \varepsilon_{t_{u+1}}
\end{aligned}$$

and for the second term we have

$$\mathbb{E}\left[\left|\Gamma^k_{t_u}(V_{t_{u+1}}1_{\mathbb{R}^D\setminus\mathcal{X}})\right|\right] \leq \mathbb{E}\left[\varepsilon_{tr}\right] = \varepsilon_{tr}.$$

For the last term we have to distinguish two cases. If we assume (GM) holds, the

operator norm yields

$$\begin{aligned}\left|\Gamma^k_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\right| &= \left|\left(\Gamma^k_{t_u} - \widehat{\Gamma}^k_{t_u}\right)\left(\widehat{V}_{t_{u+1}}\right)\right| \\ &\leq \left|\left|\Gamma^k_{t_u} - \widehat{\Gamma}^k_{t_u}\right|\right|_{op} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_\infty \\ &\leq \overline{\delta}\, \left|\left|\widehat{V}_{t_{u+1}}\right|\right|_\infty.\end{aligned}$$

Next, we consider the second case and assume that (GM*) holds. Then we have

$$\mathbb{E}\left[\left|\Gamma^k_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\right|\right] \leq \left|\left|\Gamma^k_{t_u} - \widehat{\Gamma}^k_{t_u}\right|\right|_{op} \left|\left|\widehat{V}_{t_{u+1}}\right|\right|^\star_\infty \leq \delta^\star(M)\, \left|\left|\widehat{V}_{t_{u+1}}\right|\right|^\star_\infty.$$

Hence in either case the following bound holds:

$$\mathbb{E}\left[\left|\Gamma^k_{t_u}(\widehat{V}_{t_{u+1}}) - \widehat{\Gamma}^k_{t_u}(\widehat{V}_{t_{u+1}})\right|\right] \leq \varepsilon_{gm} \max_{x\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(x)|\right]$$

with $\varepsilon_{gm} = \overline{\delta}$ if assumption (GM) holds and $\varepsilon_{gm} = \delta^\star(M)$ if assumption (GM*) holds. We need an upper bound for the maximum of the Chebyshev approximation

$$\max_{x\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(x)|\right] \leq \max_{x\in\mathcal{X}} \mathbb{E}\left[|\widehat{V}_{t_{u+1}}(x) - V_{t_{u+1}}(x)|\right] + \max_{x\in\mathcal{X}} |V_{t_{u+1}}(x)| \leq \varepsilon_{t_{u+1}} + \overline{V}_{u+1}$$

with $\overline{V}_{u+1} := \max_{x\in\mathcal{X}} |V_{t_{u+1}}(x)|$. Hence, the error bound (A.1) becomes

$$\varepsilon_{t_u} \leq \varepsilon^u_{int} + \Lambda_{\overline{N}} L_f\big((1+\varepsilon_{gm})\varepsilon_{t_{u+1}} + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\big).$$

By induction, we now show (3.5). For $u = n$ we have $\varepsilon_{t_n} \leq \varepsilon^n_{int}$ and therefore (3.5) holds for u=n. We assume that for $n, \ldots, u+1$ (3.5) holds. For the error ε_{t_u} we obtain

$$\begin{aligned}\varepsilon_{t_u} &\leq \varepsilon^u_{int} + \Lambda_{\overline{N}} L_f\big((1+\varepsilon_{gm})\varepsilon_{t_{u+1}} + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\big) \\ &\leq \varepsilon^u_{int} + \Lambda_{\overline{N}} L_f\Bigg((1+\varepsilon_{gm})\Bigg(\sum_{j=u+1}^{n} C^{j-(u+1)}\varepsilon^j_{int} \\ &\qquad + \Lambda_{\overline{N}} L_f \sum_{j=u+2}^{n} C^{j-(u+2)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j)\Bigg) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\Bigg) \\ &= \varepsilon^u_{int} + C\sum_{j=u+1}^{n} C^{j-(u+1)}\varepsilon^j_{int} + \Lambda_{\overline{N}} L_f\Bigg(C\sum_{j=u+2}^{n} C^{j-(u+2)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j) \\ &\qquad + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\Bigg) \\ &= \varepsilon^u_{int} + \sum_{j=u+1}^{n} C^{j-u}\varepsilon^j_{int} + \Lambda_{\overline{N}} L_f\Bigg(\sum_{j=u+2}^{n} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j) + \varepsilon_{tr} + \varepsilon_{gm}\overline{V}_{u+1}\Bigg) \\ &= \sum_{j=u}^{n} C^{j-u}\varepsilon^j_{int} + \Lambda_{\overline{N}} L_f \sum_{j=u+1}^{n} C^{j-(u+1)}(\varepsilon_{tr} + \varepsilon_{gm}\overline{V}_j),\end{aligned}$$

which was our claim. ∎

REFERENCES

[1] E. Bayraktar, *A proof of the smoothness of the finite time horizon American put option for jump diffusions*, SIAM J. Control Optim., 48 (2009), pp. 551–572.

[2] D. Belomestny, S. Häfner, and M. Urusov, *Regression-based complexity reduction of the nested Monte-Carlo methods*, SIAM J. Financial Math., 9 (2018), pp. 665–689.

[3] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, J. Polit. Econ., 81 (1973).

[4] M. J. Brennan and E. S. Schwartz, *The valuation of American put options*, J. Finance, 2 (1977), pp. 449–462.

[5] Y. Cai and K. L. Judd, *Shape-preserving dynamic programming*, Math. Methods Oper. Res., 77 (2013), pp. 407–421.

[6] R. Cont and P. Tankov, *Financial Modelling with Jump Processes*, Financ. Math., Chapman & Hall/CRC Press, Boca Raton, FL, 2004.

[7] V. Dominguez, I. Graham, and V. Smyshlyaev, *Stability and error estimates for Filon–Clenshaw–Curtis rules for highly oscillatory integrals*, IMA J. Numer. Anal., 31 (2011), pp. 1253–1280.

[8] T. A. Driscoll, N. Hale, and L. N. Trefethen, *Chebfun Guide*, 2014.

[9] M. Escobar, M. Mahlstedt, S. Panz, and R. Zagst, *Vulnerable exotic derivatives*, J. Derivatives, 24 (2017), pp. 84–102.

[10] F. Fang and C. W. Oosterlee, *Pricing early-exercise and discrete barrier options by Fourier-cosine series expansions*, Numer. Math., 114 (2009), p. 27.

[11] M. Gass, K. Glau, M. Mahlstedt, and M. Mair, *Chebyshev interpolation for parametric option pricing*, Finance Stoch., 22 (2018), pp. 701–731, https://doi.org/10.1007/s00780-018-0361-y.

[12] R. Geske and H. E. Johnson, *The American put option valued analytically*, J. Finance, 39 (1984), pp. 1511–1524.

[13] P. Glasserman, *Monte Carlo Methods in Financial Engineering*, Vol. 53, Springer, New York, 2003.

[14] K. Glau, R. Pachon, and C. Pötz, *Fast Calculation of Credit Exposures for Bermudan Options Using Chebyshev Interpolation*, Working paper, 2018.

[15] B. Haasdonk, J. Salomon, and B. Wohlmuth, *A reduced basis method for the simulation of American options*, in Numerical Mathematics and Advanced Applications 2011, Springer, New York, 2013, pp. 821–829.

[16] T. Haentjens and K. J. in't Hout, *ADI schemes for pricing American options under the Heston model*, Appl. Math. Finance, 22 (2015), pp. 207–237.

[17] N. Hilber, N. Reich, C. Winter, and C. Schwab, *Computational Methods for Quantitative Finance*, Springer, New York, 2013.

[18] S. Jain and C. W. Oosterlee, *The stochastic grid bundling method: Efficient pricing of Bermudan options and their Greeks*, Appl. Math. Comput., 269 (2015), pp. 412–431.

[19] K. L. Judd, *Numerical Methods in Economics*, MIT Press, Cambridge, MA, 1998.

[20] R. Kan and C. Robotti, *On moments of folded and truncated multivariate normal distributions*, J. Comput. Graph. Statist., 26 (2017), pp. 930–934.

[21] R. Korn, E. Korn, and G. Kroisandt, *Monte Carlo Methods and Models in Finance and Insurance*, CRC Press, Boca Raton, FL, 2010.

[22] S. Levendorskiĭ, *Pricing of the American put under Lévy processes*, Int. J. Theor. Appl. Finance, 7 (2004), pp. 303–335.

[23] F. A. Longstaff and E. S. Schwartz, *Valuing American options by simulation: A simple least-squares approach*, Rev. Financ. Stud., 14 (2001), pp. 113–147.

[24] R. Lord, F. Fang, F. Bervoets, and C. W. Oosterlee, *A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes*, SIAM J. Sci. Comput., 30 (2008), pp. 1678–1705.

[25] M. Mahlstedt, *Complexity Reduction for Option Pricing*, Ph.D. thesis, Technische Universität München, 2017.

[26] R. C. Merton, *Option pricing when underlying stock returns are discontinuous*, J. Financial Econ., 3 (1976), pp. 125–144.

[27] D. B. Nelson and K. Ramaswamy, *Simple binomial processes as diffusion approximations in financial models*, Rev. Financ. Stud., 3 (1990), pp. 393–430.

[28] G. Peskir and A. Shiryaev, *Optimal Stopping and Free-Boundary Problems*, Springer, New York, 2006.

[29] L. C. Rogers, *Monte Carlo valuation of American options*, Math. Finance, 12 (2002), pp. 271–286.

[30] W. Rudin, *Functional Analysis*, McGraw-Hill, New York, 1973.
[31] S. Sauter and C. Schwab, *Boundary Element Methods*, Springer Ser. Comput. Math. 39, Springer, New York, 2010.
[32] M. Schroder, *Computing the constant elasticity of variance option pricing formula*, J. Finance, 44 (1989), pp. 211–219.
[33] L. N. Trefethen, *Approximation Theory and Approximation Practice*, SIAM, Philadelphia, 2013.
[34] L. von Sydow, L. J. Höök, E. Larsson, et al., *BENCHOP–The BENCHmarking project in option pricing*, Int. J. Comput. Math., 92 (2015), pp. 2361–2379, http://www.it.uu.se/research/scientific_computing/project/compfin/benchop.