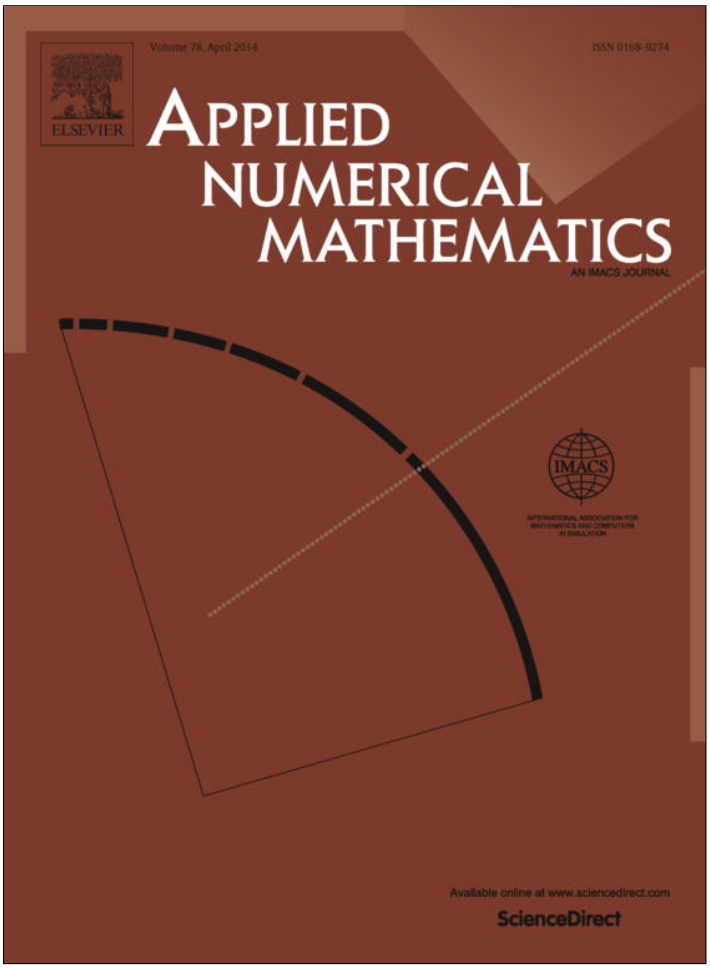
View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by CWI's Institutional Repository

Provided for non-commercial research and education use.
Not for reproduction, distribution or commercial use.



This article appeared in a journal published by Elsevier. The attached copy is furnished to the author for internal non-commercial research and education use, including for instruction at the authors institution and sharing with colleagues.

Other uses, including reproduction and distribution, or selling or licensing copies, or posting to personal, institutional or third party websites are prohibited.

In most cases authors are permitted to post their version of the article (e.g. in Word or Tex form) to their personal website or institutional repository. Authors requiring further information regarding Elsevier's archiving and manuscript policies are encouraged to visit:

http://www.elsevier.com/authorsrights

Author's personal copy

Applied Numerical Mathematics 78 (2014) 14–30

Contents lists available at ScienceDirect

Applied Numerical Mathematics

www.elsevier.com/locate/apnum

Pricing of early-exercise Asian options under Lévy processes based on Fourier cosine expansions

B. Zhang [a,*], C.W. Oosterlee [a,b]

[a] Delft University of Technology, Mekelweg 4, 2628CD, Delft, The Netherlands
[b] CWI – Centrum Wiskunde & Informatica, Amsterdam, The Netherlands

ARTICLE INFO

Article history:
Received 12 April 2013
Received in revised form 28 August 2013
Accepted 11 November 2013
Available online 6 December 2013

Keywords:
Early-exercise Asian option
Arithmetic average
Fourier cosine expansion
Clenshaw–Curtis quadrature
Exponential convergence
Graphics Processing Unit (GPU) computation

ABSTRACT

In this article, we propose a pricing method for Asian options with early-exercise features. It is based on a two-dimensional integration and a backward recursion of the Fourier coefficients, in which several numerical techniques, like Fourier cosine expansions, Clenshaw–Curtis quadrature and the Fast Fourier Transform (FFT) are employed. Rapid convergence of the pricing method is illustrated by an error analysis. Its performance is further demonstrated by various numerical examples, where we also show the power of an implementation on Graphics Processing Units (GPUs).

© 2013 IMACS. Published by Elsevier B.V. All rights reserved.

1. Introduction

An Asian option is a special type of exotic option, introduced in Japan, in 1987. Because the contract description (i.e. the pay-off function) is based on geometric or arithmetic averages of the underlying stock price at monitoring dates during the lifetime of the contract, rather than just on the present asset price, this exotic option is called *path-dependent*. The number of monitoring dates can be finite (so-called discretely-monitored Asian options) or infinite (continuously-monitored Asian options). Asian options are popular, because averages typically move in a more stable way than individual asset prices, and the volatility, inherent in asset prices, is reduced due to the averaging feature, so that Asian option holders may pay lower prices for these contracts, compared to plain European option equivalents.

There is not much information on early-exercise Asian option products in the present markets. We may encounter them in the commodity market. In the academic literature, important contributions [8,4] have been presented when pricing these Asian options by partial differential and partial integro-differential equations (PDEs and PIDEs, respectively). In [8], for example, a semi-Lagrangian time-stepping method was used to solve the P(I)DE in a time-stepping procedure. The method worked particularly well for American-style Asian options under a jump-diffusion model. As an alternative to these PIDE methods, the algorithm proposed in [1] calculates the early-exercise Asian option prices via dynamic programming, with the approximation of the option value at each time step as a piecewise-polynomial function of the underlying asset price and the average of the underlying asset prices after an appropriate change of variable. Their numerical solution and error analysis are restricted to the GBM model. However, given that we focus on the same type of Asian options (i.e. discretely-monitored, early-exercise arithmetic Asian options), we can use their results as a benchmark. Furthermore, a lattice algorithm for American Asian option pricing based on the binomial tree is presented in [7].

* Corresponding author.
E-mail addresses: Bowen.Zhang@tudelft.nl (B. Zhang), C.W.Oosterlee@cwi.nl (C.W. Oosterlee).

0168-9274/$36.00 © 2013 IMACS. Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.apnum.2013.11.004

Author's personal copy

In [16], European-style Asian options were priced by means of Fourier cosine expansions (as in the COS method [9]) and Clenshaw–Curtis quadrature. The method was named the "Asian cosine method" (in short, the *ASCOS method*). This new pricing method can be seen as an alternative to Fast Fourier Transform (FFT) and convolution methods, as in [6,11,3,12], for pricing European-style Asian options under Lévy processes.

In this paper, which is the follow-up paper of the European Asian option paper [16], we propose a robust and efficient version of the ASCOS pricing method for early-exercise Asian options, again based on Fourier expansions, Fast Fourier Transform (FFT) and Clenshaw–Curtis quadrature. The option price is calculated based on two dimensions of uncertainty, i.e. the uncertainty in the asset process, as well as in the *averaged asset process* over time. The risk-neutral formula then becomes a two-dimensional integration, based on which the continuation value is approximated at each time step. By application of the *chain rule* from probability theory, the joint conditional density function in the risk-neutral formula can be factorized into two marginal conditional density functions that are approximated by Fourier cosine expansions. To calculate the option price, we need to recursively recover the Fourier coefficients with the help of Fourier cosine expansions and Clenshaw–Curtis quadrature. The FFT is used to accelerate the algorithm.

Exponential convergence in the option price is obtained for most Lévy processes, for which we give an error analysis, combined with numerical examples. The 2D method is presented in Section 2, followed by an error analysis in Section 3. Numerical results are given in Section 4, where the efficiency and accuracy of the pricing methods are presented. Implementation has taken place on the Graphics Processing Unit (GPU). It may be interesting to see that this computer architecture improves the pricing speed drastically when pricing arithmetic Asian options with early-exercise features.

2. Early-exercise Asian options under Lévy processes

In this article the underlying asset is assumed to be an exponential function of a Lévy process, i.e. at each time t: $t > 0$, $S_t = S_0 \exp(L_t)$. The Lévy process, with initial condition $L_0 = 0$, has independent and stationary increments and is stochastically continuous. For any $s < t$, and $\forall \epsilon > 0$, we have

$$\lim_{s \to t} \mathbb{P}\big(|L_t - L_s| > \epsilon\big) = 0.$$

The (conditional) probability density function is not known for many relevant Lévy asset processes. However, its *Fourier transform*, the (conditional) characteristic function is often available, for example, by the Lévy–Khinchine theorem for Lévy processes. The ASCOS pricing algorithm is based on the Fourier transform.

An early-exercise Asian option is an early-exercise option in which the contract payoff function at each exercise date is a function of the averaged underlying asset prices, up to that date. Based on the different types of averages, Asian options can be classified into geometric and arithmetic Asian options. *Early-exercise* implies that the option may be exercised prior to the expiration date. We denote by t_0 the initial time and $\mathcal{T} = \{t_1, \ldots, t_{\mathcal{M}}\}$ be the collection of all exercise dates with $\Delta t := (t_m - t_{m-1})$, $t_0 < t_1 < \cdots < t_{\mathcal{M}} = T$. We assume that the early-exercise dates and the monitoring dates of the Asian options are the same.

In this paper, we assume that Δt is constant, but our method can be adapted straightforwardly to the case where Δt is not constant, and thus $t_m - t_{m-1}$, which may be different at different time steps, should be used in the pricing formula instead of a constant Δt.

Here, emphasis is placed on early-exercise *arithmetic* Asian options, as these are mathematically more challenging than the geometric Asian options due to the fact that the characteristic function of the arithmetic averaging process is not known, and on *fixed-strike Asian options*, with payoff functions defined by

$$g(S, t_m) = \begin{cases} \max(\frac{1}{m+1} \sum_{j=0}^{m} S_j - K, 0), & \text{for a call,} \\ \max(K - \frac{1}{m+1} \sum_{j=0}^{m} S_j, 0), & \text{for a put.} \end{cases}$$

These payoff functions change from time step to time step, due to the averaging feature.

2.1. ASCOS method for early-exercise Asian options

In this section we present a 2D pricing algorithm for early-exercise Asian options, which can be used for all Lévy processes with any number of early-exercise dates. Calculations of the continuation value and the Fourier coefficients at each time step are discussed, respectively, in Sections 2.1.1 and 2.2.

The pricing formula for an early-exercise Asian option with $\mathcal{M}$ exercise dates reads, for $m = \mathcal{M}, \mathcal{M}-1, \ldots, 2$:

$$\begin{cases} c(y_{m-1}, t_{m-1}) = e^{-r\Delta t} \int\limits_{\mathbb{R}} v(y_m, t_m) f(y_m | y_{m-1}) \, dy_m, \\ \\ v(y_{m-1}, t_{m-1}) = \max\big(g(y_{m-1}, t_{m-1}), c(y_{m-1}, t_{m-1})\big), \end{cases} \tag{1}$$

followed by

Author's personal copy

$$v(y_0,t_0)=e^{-r\Delta t}\int_{\mathbb{R}} v(y_1,t_1)f(y_1|y_0)\,dy_1. \tag{2}$$

Here, y_m is the state variable at time step t_m, and $v(x,t)$, $c(x,t)$ and $g(x,t)$ are the option value, the continuation value and the payoff at time t, respectively. $v(S,t_{\mathcal{M}})=g(S,t_{\mathcal{M}})$ is the payoff function at final time, $t_{\mathcal{M}}=T$. Function $f(y_m|y_{m-1})$ is the conditional density of y_m given y_{m-1}. Interest rate r is assumed to be constant here. In the risk-neutral formula (1) the continuation value is computed at each time step as the discounted expected value of the option price at a next time step. Moreover, to avoid arbitrage opportunities, the option value at each time step cannot be less than the payoff of the option, which is the second equation in (1).

In [9,10] the COS method was developed for the computation of continuation value $c(y_{m-1},t_{m-1})$ and option price $v(y_0,t_0)$, for Bermudan options, based on the insight that the characteristic function of the underlying Lévy asset price process is known.

2.1.1. Continuation value

At each time step, $m=\mathcal{M},\ldots,1$, we define

$$Y_m:=\frac{S_1}{S_0}+\cdots+\frac{S_m}{S_0},\qquad X_m:=\log\left(\frac{S_m}{S_0}\right),$$

and we have

$$Y_m=Y_{m-1}+e^{X_m}. \tag{3}$$

From the risk-neutral evaluation formula, where the continuation value is derived as the discounted expected option price at the next time step, we have, for $m=\mathcal{M},\ldots,1$, a two-dimensional valuation formula, as follows

$$\begin{aligned}c(y_{m-1},x_{m-1},t_{m-1})&=e^{-r\Delta t}\mathbb{E}\big(v(y_m,x_m,t_m)|y_{m-1},x_{m-1}\big)\\&=e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_m)}^{+\infty} v(y_m,x_m,t_m)f(y_m,x_m|y_{m-1},x_{m-1})\,dy_m\,dx_m,\end{aligned} \tag{4}$$

where the integration range of y_m (i.e. $y_m\geqslant\exp(x_m)$) is based on (3) and $y_{m-1}\geqslant 0$.

Truncating the integration range, gives us approximation $\hat{c}$ as,

$$\hat{c}(y_{m-1},x_{m-1},t_{m-1})=e^{-r\Delta t}\int_{a_1}^{b_1}\int_{\exp(x_m)}^{b_2} v(y_m,x_m,t_m)f(y_m,x_m|y_{m-1},x_{m-1})\,dy_m\,dx_m, \tag{5}$$

where $[a_1,b_1]$ and $[\exp(x_m),b_2]$ are the integration ranges for x_m and y_m, respectively. Integration range $[a_1,b_1]$ is calculated according to a standard formula in [10], and the calculation of b_2 will be explained in Section 2.4. By applying the *chain rule* to the joint conditional density function in (5), we find

$$\begin{aligned}f(y_m,x_m|y_{m-1},x_{m-1})&=f(y_m|x_m,y_{m-1},x_{m-1})\cdot f(x_m|y_{m-1},x_{m-1})\\&=f(y_m|x_m,y_{m-1})\cdot f(x_m|x_{m-1}).\end{aligned} \tag{6}$$

By inserting (6) into (5), the risk-neutral formula can be written as

$$\hat{c}(y_{m-1},x_{m-1},t_{m-1})=e^{-r\Delta t}\int_{a_1}^{b_1}\int_{\exp(x_m)}^{b_2} v(y_m,x_m,t_m)f(y_m|x_m,y_{m-1})\cdot f(x_m|x_{m-1})\,dy_m\,dx_m. \tag{7}$$

Whereas the conditional density function is not known analytically for many Lévy processes, the corresponding characteristic function is. Based on this, we approximate the conditional density functions by truncated Fourier cosine expansions, based on the characteristic functions, as follows,

$$\hat{f}(x_m|x_{m-1})=\frac{2}{b_1-a_1}\sum_{k=0}^{N_1-1}{}'\,Re\left(\phi_{x_m-x_{m-1}}\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\cdot\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right)\cos\left(k\pi\frac{x_m-a_1}{b_1-a_1}\right), \tag{8}$$

and

Author's personal copy

$$\hat{f}(y_m|x_m, y_{m-1}) = \sum_{j=0}^{N_2-1}{}' \frac{2}{b_2-\exp(x_m)} Re\left(\exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\right)\cos\left(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)}\right), \tag{9}$$

where $\hat{f}$ represents an approximate density function, and (9) is based on Eq. (3). For Lévy processes, defined by independent and identical increments, the (unconditional) characteristic functions of all increments of consecutive time steps, i.e. $\phi_{x_m-x_{m-1}}(u;\Delta t)$, are identical, for all time steps, and known analytically. Therefore, we use the notation $\phi(u;\Delta t) := \phi_{x_m-x_{m-1}}(u;\Delta t)$ for all time steps.

By replacing the two density functions in (7) by their approximations in (8) and (9), and interchanging the order of summation and integration, we obtain,

$$\begin{aligned}
\hat{c}(y_{m-1}, x_{m-1}, t_{m-1}) = e^{-r\Delta t}\frac{2}{b_1-a_1}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}'\int_{a_1}^{b_1}\int_{\exp(\delta_n)}^{b_2}\hat{v}(y_m,x_m,t_m)\\
\cdot Re\left(\phi\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right)\cos\left(k\pi\frac{x_m-a_1}{b_1-a_1}\right)\\
\cdot\frac{2}{b_2-\exp(x_m)}Re\left(\exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_{m-1}\right)\right)\cos\left(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)}\right)dy_m\,dx_m\\
= e^{-r\Delta t}\frac{2}{b_1-a_1}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}' Re\left(\phi\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right)\\
\cdot Re\left(\int_{a_1}^{b_1}\frac{2}{b_2-\exp(x_m)}\exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\cos\left(k\pi\frac{x_m-a_1}{b_1-a_1}\right)\right.\\
\left.\cdot\int_{\exp(\delta_n)}^{b_2}\hat{v}(y_m,x_m,t_m)\cos\left(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)}\right)dy_m\,dx_m\right).
\end{aligned} \tag{10}$$

For the integration over x_m in (10) numerical approximation is required, and Clenshaw–Curtis quadrature is employed here. The term

$$\frac{2}{b_2-\exp(x_m)}\exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\cos\left(k\pi\frac{x_m-a_1}{b_1-a_1}\right)\cos\left(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)}\right)$$

is smoothly varying[1] in x_m and the same is true for the option value, $\hat{v}(y_m,x_m,t_m)$, for all $m<\mathcal{M}$. At $t_{\mathcal{M}}=T$, $v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})$ is a function of only $y_{\mathcal{M}}$, i.e. $v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\equiv g(y_{\mathcal{M}},t_{\mathcal{M}})$. As the integrand in (10) is a smooth function of x_m at each time point, we expect an exponential convergence from the quadrature. More detailed information about Clenshaw–Curtis quadrature can be found in [5].

The Clenshaw–Curtis as well as the Gaussian quadrature rules exhibit exponential convergence for the integral under consideration (see [14]). However, Clenshaw–Curtis quadrature appears to be computationally somewhat cheaper. The integration weights for all nodes (i.e. w_n, $n=1,\ldots,n_q+2$) can be pre-computed as the Discrete Cosine Transform (DCT) of a vector of real numbers, in $O(n_q\log n_q)$ computational cost by an FFT-based algorithm for the DCT. We refer the reader to [15] for more details regarding the fast calculation of Clenshaw–Curtis quadrature rules.

Furthermore, Clenshaw–Curtis quadrature leads to nested quadrature rules, where different accuracy orders share nodes. Therefore, when studying the convergence behavior of the method, it is not necessary to re-compute certain nodes and weights with an increasing number of terms in the quadrature.

In detail, for the approximation by Clenshaw–Curtis quadrature, we have

$$\begin{aligned}
&\int_{a_1}^{b_1}\frac{2}{b_2-\exp(x_m)}\exp\left(i\frac{j\pi}{b_2-\exp(x_m)}y_{m-1}\right)\cos\left(k\pi\frac{x_m-a_1}{b_1-a_1}\right)\\
&\quad\cdot\int_{\exp(\delta_n)}^{b_2}\hat{v}(y_m,x_m,t_m)\cos\left(j\pi\frac{y_m-\exp(x_m)}{b_2-\exp(x_m)}\right)dy_m\,dx_m
\end{aligned}$$

[1] The function is continuous in x_m and so are its derivatives with respect to x_m.

Author's personal copy

$$\approx \frac{b_1-a_1}{2}\sum_{n=1}^{n_q+2} w_n \frac{2}{b_2-\exp(\delta_n)} \exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_{m-1}\right)\cos\left(k\pi\frac{\delta_n-a_1}{b_1-a_1}\right)$$
$$\cdot \int_{\exp(\delta_n)}^{b_2} \hat{v}(y_m,\delta_n,t_m)\cos\left(j\pi\frac{y_m-\exp(\delta_n)}{b_2-\exp(\delta_n)}\right)dy_m, \tag{11}$$

where

$$\delta_n = \begin{cases} \frac{b_1-a_1}{2}\cos(\frac{n\pi}{n_q})+\frac{b_1+a_1}{2}, & n=\{0,\ldots,n_q/2\},\\ \frac{a_1-b_1}{2}\cos(\frac{(n-(n_q/2+1))\pi}{n_q})+\frac{b_1+a_1}{2}, & n=\{n_q/2+1,\ldots,n_q+1\}\end{cases} \tag{12}$$

and w is an (n_q+2)-vector, defined as $w:=\{w_n\}_{n=1}^{n_q+2}=(D^Td,D^Td)^T$, with D an $(\frac{n_q}{2}+1)\times(\frac{n_q}{2}+1)$-matrix, with elements

$$D(n_1,n_2)=\frac{2}{n_q}\cos\left(\frac{(n_1-1)(n_2-1)\pi}{n_q/2}\right)\cdot\begin{cases}1/2, & n_2=\{1,n_q/2+1\},\\ 1, & \text{otherwise},\end{cases}$$

with $n_1,n_2=1,\ldots,n_q/2+1$, and vector d reads

$$d=\left(1,\frac{2}{1-4},\frac{2}{1-16},\ldots,\frac{2}{1-(n_q-2)^2},\frac{1}{1-n_q^2}\right)^T.$$

Note that the values of δ_n, w_n do not depend on k, j, m. In other words, these values only need to be calculated once and can be re-used for all k, j and for all time steps.

Inserting (11) in (10) gives us the formula for the continuation value at each time step. With the notation,

$$\hat{V}_{n,j}(t_m):=\int_{\delta_n}^{b_2}\hat{v}(y_m,\delta_n,t_m)\cos\left(j\pi\frac{y_m-\exp(\delta_n)}{b_2-\exp(\delta_n)}\right)dy_m, \tag{13}$$

and δ_n as in (12), we obtain

$$\hat{c}(y_{m-1},x_{m-1},t_{m-1})=e^{-r\Delta t}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}' Re\left(\phi\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\exp\left(ik\pi\frac{x_{m-1}-a_1}{b_1-a_1}\right)\right)$$
$$\cdot Re\left(\sum_{n=1}^{n_q+2}\frac{2}{b_2-\exp(\delta_n)}w_n\exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_{m-1}\right)\cos\left(k\pi\frac{\delta_n-a_1}{b_1-a_1}\right)\hat{V}_{n,j}(t_m)\right). \tag{14}$$

Lemma 2.1. *The computational cost to calculate the continuation value at each time step is* $O(N_1N_2n_q)$.

Proof. This can be directly seen from (14) which includes summations over N_1, N_2 and n_q. □

The 2D pricing algorithm is based on backward recursion of the Fourier coefficients $\hat{V}_{n,j}(t_m)$, defined in (13). The early-exercise Asian option price, $\hat{v}(x_0,t_0)=\hat{c}(y_0,x_0,t_0)$ is obtained by taking $m=1$ and inserting $\hat{V}_{n,j}(t_1)$ in (14). In the next subsection we will see that the $V_{n,j}(t_{\mathcal{M}})$ are known analytically. For $m=\mathcal{M}-1,\ldots,1$, coefficients $\hat{V}_{n,j}(t_m)$ can be recovered from $\hat{V}_{n,j}(t_{m+1})$.

2.2. Fourier coefficients

At maturity time, $t_{\mathcal{M}}=T$, the option value equals the payoff, so that, $\forall n, j$,

$$V_{n,j}(t_{\mathcal{M}}):=\int_{\exp(\delta_n)}^{b_2} g(y_{\mathcal{M}},t_{\mathcal{M}})\cos\left(j\pi\frac{y_{\mathcal{M}}-\exp(\delta_n)}{b_2-\exp(\delta_n)}\right)dy_{\mathcal{M}},$$

where, $\forall m=1,\ldots,\mathcal{M}$,

$$g(y_m,t_m)=\begin{cases}(\frac{S_0(1+y_m)}{m+1}-K)^+, & \text{for a call},\\ (K-\frac{S_0(1+y_m)}{m+1})^+, & \text{for a put}.\end{cases} \tag{15}$$

Author's personal copy

Thus, the Fourier coefficients at maturity read

$$V_{n,j}(t_{\mathcal{M}}) = \begin{cases} \frac{S_0}{\mathcal{M}+1}\varsigma_j(y^*_{\mathcal{M},n}, b_2) + (\frac{S_0}{\mathcal{M}+1} - K)\psi_j(y^*_{\mathcal{M},n}, b_2), & \text{for a call,} \\ (K - \frac{S_0}{\mathcal{M}+1})\psi_j(\exp(\delta_n), y^*_{\mathcal{M},n}) - \frac{S_0}{\mathcal{M}+1}\varsigma_j(\exp(\delta_n), y^*_{\mathcal{M},n}), & \text{for a put,} \end{cases} \tag{16}$$

where $y^*_{\mathcal{M},n} \equiv \frac{K(\mathcal{M}+1)}{S_0} - 1$, $\psi_j(y_l, y_u)$ is given by

$$\psi_j(y_l, y_u) := \int_{y_l}^{y_u} \cos\left(j\pi \frac{y - \delta_n}{b_2 - \delta_n}\right) dy, \tag{17}$$

and

$$\varsigma_j(y_l, y_u) = \int_{y_l}^{y_u} y \cos\left(j\pi \frac{y - \exp(\delta_n)}{b_2 - \exp(\delta_n)}\right) dy. \tag{18}$$

Both $\psi_j(y_l, y_u)$ and $\varsigma_j(y_l, y_u)$ are known analytically, and so is $V_{n,j}(t_{\mathcal{M}})$. The recursive step is presented in the following result.

Result 2.1. *For time steps t_m, $m = \mathcal{M}-1, \ldots, 1$, the continuation value, $\hat{c}(y_m, x_m, t_m)$, and the Fourier cosine coefficients, $\hat{V}_{n,j}(t_m)$, can be obtained from $\hat{V}_{n,j}(t_{m+1})$. In other words, the Fourier coefficients $\hat{V}_{n,j}(t_1)$ can be recovered recursively.*

Proof. For $m = \mathcal{M}-1, \ldots, 1$, first of all, the early-exercise points, $y^*_{m,n}$, for which $c(y^*_{m,n}, \delta_n, t_m) = g(y^*_{m,n}, t_m)$, with δ_n as in (12), need to be determined by Newton's method. The payoff function is as in (15), and the continuation value is derived via

$$\begin{aligned} \hat{c}(y_m, \delta_n, t_m) = e^{-r\Delta t} {\sum_{k=0}^{N_1-1}}' {\sum_{j=0}^{N_2-1}}' Re\left(\phi\left(\frac{k\pi}{b_1 - a_1}; \Delta t\right) \exp\left(ik\pi \frac{\delta_n - a_1}{b_1 - a_1}\right)\right) \\ \cdot Re\left(\sum_{p=1}^{n_q+2} \frac{2}{b_2 - \exp(\delta_p)} w_p \exp\left(i \frac{j\pi}{b_2 - \exp(\delta_p)} y_m\right) \cos\left(k\pi \frac{\delta_p - a_1}{b_1 - a_1}\right) \hat{V}_{p,j}(t_{m+1})\right), \end{aligned} \tag{19}$$

which is directly obtained from (14). Furthermore, the derivative of the continuation value as well as of the payoff function, with respect to y_m, can be easily computed by (19) and (15). By prescribing the initial values for the Newton iterations as $y^0_{m,n} := y^*_{m+1,n}$, with $y^0_{\mathcal{M}-1,n} := \frac{K\mathcal{M}}{S_0} - 1$ (i.e. the at-the-money at $t_{\mathcal{M}-1}$), after approximately five Newton iterations, the error in $y^*_{m,n}$ is $O(10^{-10})$. The computational cost of a Newton iteration is $O(N_1 N_2 n_q)$, which is of lower order than the calculation of Fourier cosine coefficients.

As a next step, the coefficients $\hat{V}_{n,j}(t_m)$ are split by means of these early-exercise points, as follows

$$\hat{V}_{n,j}(t_m) = \begin{cases} \hat{C}_{n,j}(\exp(\delta_n), y^*_{m,n}, t_m) + G_{n,j}(y^*_{m,n}, b_2, t_m), & \text{for a call,} \\ G_{n,j}(\exp(\delta_n), y^*_{m,n}, t_m) + \hat{C}_{n,j}(y^*_{m,n}, b_2, t_m), & \text{for a put,} \end{cases} \tag{20}$$

where $\hat{C}_{n,j}$, $G_{n,j}$ are Fourier cosine coefficients of the continuation value and payoff at t_m, respectively. Coefficient $G_{n,j}$ is found analytically,

$$G_{n,j}(t_m) = \begin{cases} \frac{S_0}{m+1}\varsigma_j(y^*_{m,n}, b_2) + (\frac{S_0}{m+1} - K)\psi_j(y^*_{m,n}, b_2), & \text{for a call,} \\ (K - \frac{S_0}{m+1})\psi_j(\exp(\delta_n), y^*_{m,n}) - \frac{S_0}{m+1}\varsigma_j(\exp(\delta_n), y^*_{m,n}), & \text{for a put,} \end{cases} \tag{21}$$

and coefficient $\hat{C}_k$, defined by

$$\hat{C}_{n,j}(y_l, y_u, t_m) = \int_{y_l}^{y_u} \hat{c}(y_m, \delta_n, t_m) \cos\left(j\pi \frac{y_m - \exp(\delta_n)}{b_2 - \exp(\delta_n)}\right) dy_m, \tag{22}$$

where integration range $[y_l, y_u] \in [\delta_n, b_2]$ is computed numerically.

Author's personal copy

By substituting for $\hat{c}(y_m, \delta_n, t_m)$ in (22) its expression in (19) and interchanging integration and summation, we obtain

$$\hat{C}_{n,j}(y_l, y_u, t_m) = e^{-r\Delta t} \sum_{k=0}^{N_1-1}{}' \sum_{l=0}^{N_2-1}{}' Re\left(\phi\left(\frac{k\pi}{b_1-a_1}; \Delta t\right) \exp\left(ik\pi \frac{\delta_n - a_1}{b_1-a_1}\right)\right)$$
$$\cdot Re\left(\sum_{p=1}^{n_q+2} \Lambda(k,l,p) \int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos\left(j\pi \frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right) dy_m\right), \tag{23}$$

where

$$\Lambda(k,l,p) := \frac{2}{b_2-\exp(\delta_p)} w_p \cos\left(k\pi \frac{\delta_p - a_1}{b_1-a_1}\right) \hat{V}_{p,l}(t_{m+1}). \tag{24}$$

The integral in (23) is known analytically. We have, $\forall y_l, y_u, l, j,\ j, l = 0, \ldots, N_2-1,\ j \neq l$,

$$\int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos\left(j\pi \frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right) dy_m$$
$$= \frac{1}{j^2-l^2} \frac{d-c}{\pi} \left(\exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_u\right) \sin\left(j\pi \frac{y_u - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right)\right.$$
$$- \exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_l\right) \sin\left(j\pi \frac{y_l - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right)$$
$$\left. + il\left(\exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_u\right) \cos\left(j\pi \frac{y_u - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right) - \exp\left(\frac{il\pi}{b_2-\exp(\delta_p)} y_l\right) \cdot \cos\left(j\pi \frac{y_l - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right)\right)\right),$$

and, if $j = l,\ j \neq 0,\ l \neq 0$,

$$\int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos\left(j\pi \frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right) dy_m$$
$$= \frac{\exp(il\pi \frac{\exp(\delta_p)}{b_2-\exp(\delta_p)})}{2} (y_u - y_l) + \left(-\frac{i}{\pi}\right) \frac{b_2-\exp(\delta_p)}{2} \exp\left(il\pi \frac{\exp(\delta_p)}{b_2-\exp(\delta_p)}\right)$$
$$\cdot \frac{\exp(i(j+l)\frac{y_u-\exp(\delta_p)}{b_2-\exp(\delta_p)}\pi) - \exp(i(j+l)\frac{y_l-\exp(\delta_n)}{b_2-\exp(\delta_p)}\pi)}{j+l},$$

and, finally, for $l = j = 0$,

$$\int_{y_l}^{y_u} \exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)} y_m\right) \cos\left(j\pi \frac{y_m - \exp(\delta_p)}{b_2-\exp(\delta_p)}\right) dy_m = y_u - y_l.$$

Fourier coefficients $\hat{C}_{n,j}(y_l, y_u, t_m)$ can thus be calculated directly from (23) without additional numerical techniques.

From (19) and (23) we see that the continuation value as well as the Fourier coefficients at t_m, $m = \mathcal{M}-1, \ldots, 1$, can be recovered from the Fourier coefficients at t_{m+1}. Coefficients $V_{n,j}(t_1)$, $\forall n, j$, are recovered at the end of the backward recursion. □

The value of the Asian option with early-exercise features is then obtained by inserting $V_{n,j}(t_1)$ into (14).

2.3. Computational cost and Fast Fourier Transform

Lemma 2.2. *With the use of the Fast Fourier Transform (FFT) the computational cost of the 2D ASCOS method for pricing early-exercise Asian options with $\mathcal{M}$ early-exercise dates is $O((\mathcal{M}-1)N_1N_2 \log_2 N_2 n_q)$.*

Proof. Newton's method is applied to determine the $y^*_{m,n}$-values with $n = 1, \ldots, n_q+2$, for which the continuation value $\hat{c}(y_m, \delta_n, t_m)$ must be computed by (19). Term

$$Re\left(\sum_{p=1}^{n_q+2} \frac{2}{b_2-\exp(\delta_p)} w_p \exp\left(i\frac{j\pi}{b_2-\exp(\delta_p)} y_m\right) \cos\left(k\pi \frac{\delta_p - a_1}{b_1-a_1}\right) \hat{V}_{p,j}(t_{m+1})\right)$$

Author's personal copy

in (19) is calculated once and can be reused in all iteration steps and for all δ_n-values. Therefore, we perform $O(N_1N_2n_q)$ computations to determine $y^*_{m,1}$, and to compute $y^*_{m,n}$, $n=2,\ldots,n_q+2$, only $O(N_1N_2)$ computations are needed. We end up with $O(N_1N_2n_q)$ computations to determine all early-exercise points.

To compute $\hat{C}_{n,j}(y_l,y_u,t_m)$ at each time step we perform $O(N_1N_2n_q)$ computations, as the integration in (23) has an analytically known solution. We need to calculate $\hat{C}_{n,j}(y_l,y_u,t_m)$ for each value of n and j. Term

$$Re\left(\sum_{p=1}^{n_q+2}\Lambda(k,l,p)\int_{y_l}^{y_u}\exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)}y_m\right)\cos\left(j\pi\frac{y_m-\exp(\delta_p)}{b_2-\exp(\delta_p)}\right)dy_m\right) \tag{25}$$

in (23) need not be re-computed for different n, and we have $O(N_1N_2n_q)$ computations in total for all n, with $n=1,\ldots,n_q+2$. To determine the Fourier coefficients, $\hat{C}_{n,j}(y_l,y_u,t_m)$, with $j=0,\ldots,N_2-1$, we require $O(N_1N_2^2n_q)$ computations, at each time step.

We need to repeat all computations for time steps t_m, $m=\mathcal{M}-1,\ldots,1$, so that the overall computational cost for the pricing technique is $O((\mathcal{M}-1)N_1N_2^2n_q)$.

The Fast Fourier Transform (FFT) can however be employed to reduce this computational cost. Eq. (23) can be rewritten, $\forall k,p$, as

$$\begin{aligned}\hat{C}^{k,p}_{n,j}=e^{-r\Delta t}{\sum_{l=0}^{N_2-1}}'Re\left(\phi\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\exp\left(ik\pi\frac{\delta_n-a_1}{b_1-a_1}\right)\right)\\ \cdot Re\left(\Lambda(k,l,p)\int_{y_l}^{y_u}\exp\left(i\frac{l\pi}{b_2-\exp(\delta_p)}y_m\right)\cos\left(j\pi\frac{y_m-\exp(\delta_p)}{b_2-\exp(\delta_p)}\right)dy_m\right).\end{aligned} \tag{26}$$

If we use the vector notation $\hat{\mathbf{C}}^{k,p}_n:=\{\hat{C}^{k,p}_{n,j}\}_{j=0}^{N_2-1}$, we can write

$$\begin{aligned}\hat{\mathbf{C}}^{k,p}_n=\frac{e^{-r\Delta t}}{\pi}Re\left(\phi\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\exp\left(ik\pi\frac{\exp(\delta_n)-a_1}{b_1-a_1}\right)\right)\\ \cdot Im\Big(\big(H^c(y_l,y_u,p)+H^s(y_l,y_u,p)\big)\mathbf{u}(k,p)\Big),\end{aligned} \tag{27}$$

where $Im(\cdot)$ denotes taking the imaginary part of the input argument, and $\mathbf{u}$ is a vector with elements

$$\mathbf{u}_l(k,p):=\Lambda(k,l,p)\exp\left(\frac{il\pi\exp(\delta_p)}{b_2-\exp(\delta_p)}\right).$$

Moreover, H^c and H^s have a Hankel and Toeplitz structure, respectively, with elements as follows,

$$H^c_{j,l}(x_1,x_2,p)=\begin{cases}\frac{(x_2-x_1)\pi i}{b_2-\exp(\delta_p)}, & \text{if } j=l=0,\\ \frac{1}{(l+j)}[\exp(\frac{((l+j)x_2-(l+j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)})-\exp(\frac{((l+j)x_1-(l+j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)})], & \text{otherwise,}\end{cases} \tag{28}$$

and

$$H^s_{j,l}(x_1,x_2,p)=\begin{cases}\frac{(x_2-x_1)\pi i}{b_2-\exp(\delta_p)}, & \text{if } j=l=0,\\ \frac{1}{(l-j)}[\exp(\frac{((l-j)x_2-(l-j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)})-\exp(\frac{((l-j)x_1-(l-j)\exp(\delta_p))\pi i}{b_2-\exp(\delta_p)})], & \text{otherwise.}\end{cases} \tag{29}$$

As pointed out in [10] the FFT can be used to calculate matrix-vector multiplications in (27) due to the Hankel and Toeplitz structure.

To compute vector $\hat{\mathbf{C}}^{k,p}_n$ for each pair of (k,p), with $k=0,\ldots,N_1-1$, $p=1,\ldots,n_q+2$, $O(N_2\log_2N_2)$ computations are performed, so that we require $O(N_1N_2\log_2N_2n_q)$ computations to compute $\hat{\mathbf{C}}^{k,p}_n$ for all k,p. Term $Im((H^c+H^s)\mathbf{u})$ can be reused for all $n=1,\ldots,n_q+2$, and $O(N_2\log_2N_2)$ computations are needed for all Fourier coefficients.

At the final stage of the algorithm, we need to add up all $k\cdot p$ elements, i.e.

$$\hat{C}_{n,j}(y_l,y_u,t_m)={\sum_{k=0}^{N_1-1}}'\sum_{p=1}^{n_q+2}\hat{C}^{k,q}_{n,j}(y_l,y_u,t_m), \tag{30}$$

with $\hat{C}^{k,q}_{n,j}(y_l,y_u,t_m)$ defined in (26) and computed by (27).

Author's personal copy

Denoting, based on (27), that

$$A_1(k,n) := \frac{e^{-r\Delta t}}{\pi} Re\left(\phi\left(\frac{k\pi}{b_1-a_1}; \Delta t\right) \exp\left(ik\pi \frac{\exp(\delta_n)-a_1}{b_1-a_1}\right)\right),$$
$$A_2(k,p) := Im\big(\big(H^c(y_l,y_u,p) + H^s(y_l,y_u,p)\big)\mathbf{u}(k,p)\big),$$

Eq. (30) can be computed efficiently, as summarized in the algorithm below.

Algorithm. Efficient computation of (30)

- Step 1: For $k = 0,\ldots,N_1-1$, compute
$$A_2(k) := \sum_{p=1}^{n_q+2} A_2(k,p).$$
- Step 2: For $n = 1,\ldots,n_q+2$, compute
$$\hat{C}_n := \{\hat{\mathbf{C}}_{n,j}\}_{j=0}^{N_2-1} = \sum_{k=0}^{N_1-1}{}' A_1(k,n)\cdot A_2(k).$$

For each vector $A_2(k,p)$, we perform $O(N_2\log_2 N_2)$ computations, resulting in $O(N_1N_2\log_2 N_2 n_q)$ computations for Step 1. In Step 2, the computational cost is $O(N_1N_2n_q)$. In total, $O(N_1N_2\log_2 N_2 n_q)$ computations are needed at each time step to compute all Fourier coefficients defined in (30). Therefore, by the use of the FFT, the computational cost of the complete algorithm is reduced to $O((\mathcal{M}-1)N_1N_2\log_2 N_2 n_q)$. □

The overall 2D ASCOS pricing algorithm is summarized below.

ASCOS Algorithm. Pricing early-exercise arithmetic Asian options.

Initialization

- For $n = 1,\ldots,n_q+2$, $j = 0,\ldots,N_2-1$, compute $V_{n,j}(t_{\mathcal{M}})$ from (16).

Main Loop to Recover $\hat{V}_{n,j}(t_m)$: For $m = \mathcal{M}-1$ to 1,

- Determine the early-exercise points, $y^*_{m,n}$, for $n = 1,\ldots,n_q+2$, with $\hat{c}(y^*_{m,n},\delta_n,t_m) = g(y^*_{m,n},t_m)$, by Newton's method. Continuation value and payoff function are given by (19) and (15), respectively.
- Compute the Fourier coefficients $\hat{V}_{n,j}(t_m)$.
  - For $k = 0,\ldots,N_1-1$, compute each column of matrix $\hat{\mathbf{C}}^k_n := \{\hat{C}^k_{n,j}\}_{j=0}^{N_2}$ by (27) with the help of the Fast Fourier Transform.
  - Compute $\hat{C}_{n,j}(t_m)$, $\forall n, j$, from (30).
  - Compute $G_{n,j}(t_m)$, $\forall n, j$, from (21).
  - Calculate the Fourier coefficients $\hat{V}_{n,j}(t_m)$ by inserting $\hat{C}_{n,j}(t_m)$ and $G_{n,j}(t_m)$ into (20).

Final step:

- Compute the early-exercise Asian option value, $\hat{v}(x_0,t_0)$, by inserting $\hat{V}_{n,j}(t_1)$ in (14).

2.4. Integration range of Y_m

Here, we explain how to determine the upper bound b_2 in (5), so that the truncation error in Y_m (defined as $\sum_{j=1}^{m}\frac{S_j}{S_0}$), with integration range $[\exp(x_m), b_2]$ can be controlled. In [9,10], a suitable integration range for a random variable X was given by means of the cumulants, as follows

$$[\ell,\upsilon] := \left[\xi_1(X) - L\sqrt{\xi_2(X)+\sqrt{\xi_4(X)}}, \xi_1(X) + L\sqrt{\xi_2(X)+\sqrt{\xi_4(X)}}\right]. \tag{31}$$

Author's personal copy

By $\xi_n(X)$, we denote the nth cumulant of X, computed via

$$\xi_n(X) := \frac{1}{i^n} \frac{\partial^n (t\Phi(\mathbf{u}))}{\partial \mathbf{u}^n} \bigg| \mathbf{u} = 0,$$

where $t\Phi(\mathbf{u})$ is the exponent of the characteristic function, $\phi(\mathbf{u}; t)$, i.e.

$$\phi(\mathbf{u}; t) = e^{t\Phi(\mathbf{u})}.$$

For arithmetic Asian options, it is however expensive to calculate these cumulants for Y_m, and therefore we propose other integration range boundaries, that are very similar to those in (31). For Y_m, $\forall m = 1, \ldots, \mathcal{M}$, we have

$$\xi_1\left(m\frac{S_1}{S_0}\right) \leqslant \xi_1(Y_m) \leqslant \xi_1\left(m\frac{S_m}{S_0}\right),$$
$$0 \leqslant \xi_2(Y_m) \leqslant \xi_2\left(m\frac{S_m}{S_0}\right), \quad 0 \leqslant \xi_4(Y_m) \leqslant \xi_4\left(m\frac{S_m}{S_0}\right).$$

The upper bound of Y_m can thus be defined as

$$b_2 := \xi_1\left(m\frac{S_m}{S_0}\right) + L\sqrt{\xi_2\left(m\frac{S_m}{S_0}\right) + \sqrt{\xi_4\left(m\frac{S_m}{S_0}\right)}}. \tag{32}$$

For general exponential Lévy processes, we will however use $\exp(\xi_n(\log(m\frac{S_m}{S_0})))$ instead of $\xi_n(m\frac{S_m}{S_0})$, which gives us a slightly different range of integration, because $\log(\xi_n(m\frac{S_m}{S_0})) \neq \xi_n(\log(m\frac{S_m}{S_0}))$. However, the cumulants for $\log(m\frac{S_m}{S_0})$ are known in closed form because of the properties of Lévy processes. For $n = 1$, $\xi_1(\log(m\frac{S_m}{S_0})) = \log(m) + m\xi_1(R)$, and, for $n \geqslant 2$, $\xi_n(\log(m\frac{S_m}{S_0})) = m\xi_n(R)$, where R denotes the logarithm of the increment between any two consecutive time steps of a Lévy process.

Setting $L = 10$ results in highly accurate option prices for most Lévy processes [9,10]. With a wider integration range, the truncation error will be smaller, but an increasing number of Fourier cosine terms need to be used (which makes computations more costly). A larger value for L is recommended when the time to maturity is small ($T < 0.1$), or when the underlying Lévy process has a fat tailed distribution.

3. Error analysis

Here, we give a detailed error analysis of the 2D ASCOS method for the early-exercise arithmetic Asian options from Section 2. We identify three different types of errors, for which we first introduce some notation.

The *truncation error*, ϵ_T, for any random variable, Z, with integration range $[a, b]$, is defined as

$$\epsilon_T\big(Z; [a, b]\big) := \int_{\mathbf{R}\setminus[a,b]} f_Z(z)\, dz, \tag{33}$$

and it decreases as the integration range $[a, b]$ increases. In other words, for a sufficiently large integration range, this error will not dominate the total error in the arithmetic Asian option price.

For Y_m we truncate the integration range at one side, so that the truncation error is here defined as

$$\epsilon_T(Y_m; b_2) := \int_{b_2}^{+\infty} f_{Y_m}(y)\, dy. \tag{34}$$

The *error due to the number of terms used in the Fourier cosine expansion* is denoted by ϵ_F. We know, from [5,9], that for $f_Z(z) \in \mathbf{C}^\infty[a, b]$, this error can be bounded by

$$\big|\epsilon_F(Z; N)\big| \leqslant P^*(N) \exp\big(-(N-1)\nu_F\big), \tag{35}$$

with $\nu_F > 0$ a constant and a term $P^*(N)$, which varies less than exponentially with respect to N. Note that, although the upper bound of ϵ_F is not an explicit function of the underlying state variable Z, the state variable is connected to the smoothness of the density function which influences the convergence behavior.

Author's personal copy

When the probability density function has a discontinuous derivative, the error can be bounded by

$$\left|\epsilon_F(Z;N)\right| \leqslant \frac{\bar{P}^*(N)}{(N-1)^{\beta-1}},$$

where $\bar{P}^*(N)$ is a constant and $\beta \geqslant 1$.

Error ϵ_F thus decays either exponentially with respect to N, if the density function $f(z) \in \mathbf{C}^\infty[a,b]$, or otherwise algebraically.

We denote the error from the Clenshaw–Curtis quadrature (11) by ϵ_q. For integrands belonging to $C^\infty[a,b]$, which is the case here, the Clenshaw–Curtis related error ϵ_q decays exponentially, i.e.,

$$\left|\epsilon_q(n_q)\right| \leqslant P(n_q)\exp\left(-(n_q-1)\nu_q\right), \tag{36}$$

with $\nu_q > 0$ constant and a term $P(n_q)$, which varies less than exponentially with respect to n_q.

It can be proved recursively, that if $f(x_j)$, $j=1,\ldots,\mathcal{M}$, is smooth then $f(y_j)$, $j=1,\ldots,\mathcal{M}$, with $y_j = \sum_{i=1}^{j}\exp(x_i)$ is also smooth, so that averaging does not influence the convergence speed negatively.

We denote by $\epsilon(\hat{c}(y_m,x_m,t_m))$, $\epsilon(V_{n,j}(t_m))$ and $\epsilon^*_{m,n}$, the errors in the continuation value, in the Fourier coefficients and in the early-exercise points, $y^*_{m,n}$, at time step t_m, respectively.

We use the notation $\epsilon_{cos}(Z)$, $\forall Z$ to denote the error of one step of the COS method [9], in which the integration range of Z is truncated and the conditional density function of Z is approximated by the characteristic function via Fourier cosine expansions. Therefore we have that, $\forall m$,

$$\epsilon_{cos}(X_m) := \int_{\mathbb{R}} v(y_m,x_m,t_m) f(x_m|x_{m-1})\,dx_m - \int_{a_1}^{b_1} v(y_m,x_m,t_m)\hat{f}(x_m|x_{m-1})\,dx_m, \tag{37}$$

and

$$\epsilon_{cos}(Y_m) := \int_{\exp(x_m)}^{+\infty} v(y_m,x_m,t_m) f(y_m|x_m,y_{m-1})\,dy_m - \int_{\exp(x_m)}^{b_2} v(y_m,x_m,t_m)\hat{f}(y_m|x_m,y_{m-1})\,dy_m. \tag{38}$$

In [9] an error analysis for the COS method was given where the truncation error and the error due to Fourier cosine expansions were analyzed, and it is concluded that $\forall Z,a,b,N$,

$$\epsilon_{cos}(Z) = O\left(\epsilon_T\left(Z;[a,b]\right)\right) + \epsilon_F(Z;N), \tag{39}$$

with ϵ_T and ϵ_F defined in (33) and (35), respectively. In the common notation $\epsilon(x) = O(\varsigma)$, $\forall x \in \mathbb{R}$, which means that there exists $Q > 0$, so that $|\epsilon(x)| \leqslant Q|\varsigma|$.

Our error analysis is based on backward recursion, i.e. first of all we analyze the error in the continuation value, $\hat{c}(y_{\mathcal{M}-1},x_{\mathcal{M}-1},t_{\mathcal{M}-1})$, in Section 3.1, after which the error propagation in the time steps $t_m, m=\mathcal{M}-2,\ldots,1$, are discussed in Section 3.2.

3.1. Initial error

In this subsection, the errors arising in Eqs. (4) to (14) are discussed. We denote by ϵ_1 the error from Eqs. (4) to (10), and by ϵ_2 the error from Eqs. (10) to (14). Therefore $\epsilon(\hat{c}(y_{\mathcal{M}-1},x_{\mathcal{M}-1},t_{\mathcal{M}-1})) = \epsilon_1+\epsilon_2$.

For ϵ_1 we have

$$\begin{aligned}\epsilon_1 = {} & e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_{\mathcal{M}})}^{+\infty} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}}) f(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1}) f(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dy_{\mathcal{M}}\,dx_{\mathcal{M}} \\ & - e^{-r\Delta t}\int_{a_1}^{b_1}\int_{\exp(x_{\mathcal{M}})}^{b_2} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})\cdot\hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dy_{\mathcal{M}}\,dx_{\mathcal{M}},\end{aligned}$$

where $\hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1})$ and $\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})$ are defined in (8) and (9), respectively.

Author's personal copy

Error ϵ_1 can be written as

$$\begin{aligned}
\epsilon_1 &= e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_{\mathcal{M}})}^{+\infty} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})f(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})\,dy_{\mathcal{M}}f(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\\
&\quad - e^{-r\Delta t}\int_{\mathbb{R}}\int_{\exp(x_{\mathcal{M}})}^{b_2} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})\,dy_{\mathcal{M}}f(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\\
&\quad + e^{-r\Delta t}\int_{\exp(x_{\mathcal{M}})}^{b_2}\int_{\mathbb{R}} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})f(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\,dy_{\mathcal{M}}\\
&\quad - e^{-r\Delta t}\int_{\exp(x_{\mathcal{M}})}^{b_2}\int_{a_1}^{b_1} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})\hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\,dy_{\mathcal{M}}\\
&= e^{-r\Delta t}\epsilon_{cos}(Y_{\mathcal{M}})\\
&\quad + e^{-r\Delta t}\int_{\exp(x_{\mathcal{M}})}^{b_2}\int_{\mathbb{R}} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})f(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\,dy_{\mathcal{M}}\\
&\quad - e^{-r\Delta t}\int_{\exp(x_{\mathcal{M}})}^{b_2}\int_{a_1}^{b_1} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})\hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\,dy_{\mathcal{M}},
\end{aligned}\tag{40}$$

where $\epsilon_{cos}(Y_{\mathcal{M}})$ is defined in (38).

Furthermore, with $\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})$ defined in (9), we have that, $\forall y_{\mathcal{M}}\in[\exp(x_{\mathcal{M}}),b_2]$,

$$\left|\hat{f}(y_{\mathcal{M}}|x_{\mathcal{M}},y_{\mathcal{M}-1})\right| \leqslant \frac{2}{b_2-\exp(x_{\mathcal{M}})}N_2 \leqslant \frac{2}{b_2-\exp(a_1)}N_2.$$

Then

$$\begin{aligned}
|\epsilon_1| &\leqslant e^{-r\Delta t}\left|\epsilon_{cos}(Y_{\mathcal{M}})\right| + e^{-r\Delta t}\frac{2N_2}{b_2-\exp(a_1)}\left|\int_{\exp(x_{\mathcal{M}})}^{b_2}\left(\int_{\mathbb{R}} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})f(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\right.\right.\\
&\quad \left.\left. - \int_{a_1}^{b_1} v(y_{\mathcal{M}},x_{\mathcal{M}},t_{\mathcal{M}})\hat{f}(x_{\mathcal{M}}|x_{\mathcal{M}-1})\,dx_{\mathcal{M}}\right)dy_{\mathcal{M}}\right|\\
&\leqslant e^{-r\Delta t}\left|\epsilon_{cos}(Y_{\mathcal{M}})\right| + e^{-r\Delta t}2N_2\left|\epsilon_{cos}(X_{\mathcal{M}})\right|
\end{aligned}\tag{41}$$

where $\epsilon_{cos}(X_{\mathcal{M}})$ is defined in (37).

From (39) we find

$$\epsilon_1 = O\left(N_2\left(\epsilon_T\left(X_{\mathcal{M}};[a_1,b_1]\right)+\epsilon_F(X_{\mathcal{M}};N_1)\right)+\epsilon_T(Y_{\mathcal{M}};b_2)+\epsilon_F(Y_{\mathcal{M}};N_2)\right),\tag{42}$$

which is the error made in the steps up to Eq. (10).

At $t_{\mathcal{M}-1}$, the Fourier coefficients of the option value, $V_{n,j}(t_{\mathcal{M}})$ are known analytically. Therefore, the error from Eq. (10) to Eq. (14) is only due to approximation (11), where the Clenshaw–Curtis quadrature was used. For each j,k the error in the numerical quadrature is $O(\epsilon_q(n_q))$, with ϵ_q defined in (36). Thus, $\epsilon_2 = O(N_1N_2\epsilon_q(n_q))$.

Summarizing, the error in $\hat{c}(y_{\mathcal{M}-1},x_{\mathcal{M}-1},t_{\mathcal{M}-1})$, which is the sum of ϵ_1 (42) and ϵ_2, is found to be

$$\begin{aligned}
&\epsilon\left(\hat{c}(y_{\mathcal{M}-1},x_{\mathcal{M}-1},t_{\mathcal{M}-1})\right)\\
&\quad = O\left(N_2\left(\epsilon_T\left(X_{\mathcal{M}};[a_1,b_1]\right)+\epsilon_F(X_{\mathcal{M}};N_1)\right)+\epsilon_T(Y_{\mathcal{M}};b_2)+\epsilon_F(Y_{\mathcal{M}};N_2)+N_1N_2\epsilon_q(n_q)\right).
\end{aligned}\tag{43}$$

With integration ranges $[a_1,b_1]$ and b_2 carefully chosen, truncation errors $\epsilon_T(X_{\mathcal{M}};[a_1,b_1])$ and $\epsilon_T(Y_{\mathcal{M}};b_2)$ will not be the dominant parts of error (43). If the increments of the underlying Lévy process are governed by smooth density functions, then, $\forall m$, the density function of $X_m=\log(S_m/S_0)$ is smooth, and so is the density function of $Y_m, m=1,\ldots,\mathcal{M}$, which can be proved using (3). Then, the error in the continuation value decays to zero exponentially, with respect to N_1,N_2,n_q.

Author's personal copy

Inserting (35) and (36) into (43), gives us

$$\begin{aligned}&\left|\epsilon\left(\hat{c}(y_{\mathcal{M}-1},x_{\mathcal{M}-1},t_{\mathcal{M}-1})\right)\right|\\&\quad\leqslant P^*(N_1,N_2,n_q)\left(\exp\left(-(N_1-1)\nu_1\right)+\exp\left(-(N_2-1)\nu_2\right)+\exp\left(-(n_q-1)\nu_q\right)\right),\end{aligned}\tag{44}$$

where $P^*(N_1,N_2,n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q, so that the error in (44) decays exponentially with respect to N_1, N_2, n_q.

If the density function of Lévy increments is not smooth, then the error in the Fourier cosine expansion decays algebraically with respect to N_1 and N_2, but the error due to the use of Clenshaw–Curtis quadrature still decays exponentially with respect to n_q.

3.2. Error propagation

Regarding the propagation of the error through time, we state the following lemma:

Lemma 3.1 *(Error propagation). For $m=\mathcal{M}-2,\ldots,0$, assuming that at time step t_{m+1}, $\forall y_{m+1},x_{m+1},\exists P(N_1,N_2,n_q)$, so that,*

$$\begin{aligned}&\left|\epsilon\left(\hat{c}(y_{m+1},x_{m+1},t_{m+1})\right)\right|\\&\quad\leqslant P(N_1,N_2,n_q)\left(\exp\left(-(N_1-1)\nu_1\right)+\exp\left(-(N_2-1)\nu_2\right)+\exp\left(-(n_q-1)\nu_q\right)\right),\end{aligned}\tag{45}$$

where $P(N_1,N_2,n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q, then, at time step t_m, we can show $\forall y_m,x_m,\exists\bar{P}(N_1,N_2,n_q)$, that

$$\begin{aligned}&\left|\epsilon\left(\hat{c}(y_m,x_m,t_m)\right)\right|\\&\quad\leqslant \bar{P}(N_1,N_2,n_q)\left(\exp\left(-(N_1-1)\nu_1\right)+\exp\left(-(N_2-1)\nu_2\right)+\exp\left(-(n_q-1)\nu_q\right)\right),\end{aligned}\tag{46}$$

where $\bar{P}(N_1,N_2,n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q.

Proof. This is a proof based on mathematical induction.

First, we compute the error in the Fourier coefficients, $\hat{V}_{n,j}(t_{m+1})$, after which we analyze the error in $\hat{c}(y_m,x_m,t_m)$.

Error $\epsilon(\hat{V}_{n,j}(t_{m+1}))$ consists of two parts, the error in the Fourier cosine coefficients of the continuation value, and the error due to an incorrect value of the early-exercise point. We take as example a call option with a positive-valued error in the early-exercise points. The analysis of the error propagation for other cases (negatively-valued error, put option) goes similarly. For a call option, with $\epsilon^*_{m+1,n}>0$, we have

$$\begin{aligned}\epsilon\left(\hat{V}_{n,j}(t_{m+1})\right)&=\left(C_{n,j}\left(\exp(\delta_n),y^*_{m+1,n},t_{m+1}\right)-\hat{C}_{n,j}\left(\exp(\delta_n),y^*_{m+1,n},t_{m+1}\right)\right)\\&\quad+\left(G_{n,j}\left(y^*_{m+1,n},y^*_{m+1,n}+\epsilon^*_{m+1,n},t_{m+1}\right)-\hat{C}_{n,j}\left(y^*_{m+1,n},y^*_{m+1,n}+\epsilon^*_{m+1,n},t_{m+1}\right)\right)\\&=\int_{\exp(\delta_n)}^{y^*_{m+1,n}}\epsilon\left(\hat{c}(y_{m+1},\delta_n,t_{m+1})\right)\cos\left(j\pi\frac{y_{m+1}-\exp(\delta_n)}{b_2-\exp(\delta_n)}\right)dy_{m+1}\\&\quad+\int_{y^*_{m+1,n}}^{y^*_{m+1,n}+\epsilon^*_{m+1,n}}\left(g(y_{m+1},t_{m+1})-\hat{c}(y_{m+1},\delta_n,t_{m+1})\right)\cos\left(j\pi\frac{y_{m+1}-\exp(\delta_n)}{b_2-\exp(\delta_n)}\right)dy_{m+1},\end{aligned}\tag{47}$$

with $g(y_{m+1},t_{m+1})$ defined in (15).

The error in the continuation value $\hat{c}(y_m,x_m,t_m)$ is composed of two parts. The first part is the error in which $\epsilon(\hat{V}_{n,j}(t_{m+1}))$ has not yet been considered and the error comes solely from the same numerical methods used to compute the continuation value in the first recursion step. The second part of the error is the additional error with $\epsilon(\hat{V}_{n,j}(t_{m+1}))$ taken into consideration.

Based on the arguments in Section 3.1, we find

$$\begin{aligned}\epsilon\left(\hat{c}(y_m,x_m,t_m)\right)&=O\left(N_2\left(\epsilon_T\left(X_{m+1};[a_1,b_1]\right)+\epsilon_F(X_{m+1};N_1)\right)\right.\\&\quad\left.+\epsilon_T(Y_{m+1};b_2)+\epsilon_F(Y_{m+1};N_2)+N_1N_2\epsilon_q(n_q)\right)\\&\quad+e^{-r\Delta t}\sum_{k=0}^{N_1-1}{}'\sum_{j=0}^{N_2-1}{}'Re\left(\phi\left(\frac{k\pi}{b_1-a_1};\Delta t\right)\exp\left(ik\pi\frac{x_m-a_1}{b_1-a_1}\right)\right)\\&\quad\cdot Re\left(\sum_{n=1}^{n_q+2}\frac{2}{b_2-\exp(\delta_n)}w_n\exp\left(i\frac{j\pi}{b_2-\exp(\delta_n)}y_m\right)\cos\left(k\pi\frac{\delta_n-a_1}{b_1-a_1}\right)\epsilon\left(\hat{V}_{n,j}(t_{m+1})\right)\right).\end{aligned}\tag{48}$$

Author's personal copy

To analyze these errors, we define a European option, v_α, from t_m to t_{m+1}, with payoff function

$$v_\alpha(y_{m+1}, x_{m+1}, t_{m+1}, L_1, L_2) := \begin{cases} 1, & \text{if} \quad y_{m+1} \in [L_1, L_2], \\ 0, & \text{otherwise.} \end{cases}$$

Then, the option value at t_m, $\forall L_1, L_2 \in [\exp(x_{m+1}), +\infty]$, can be written as

$$\begin{aligned} v_\alpha(y_m, x_m, t_m, L_1, L_2) &= e^{-r\Delta t} \int_{\mathbb{R}} \int_{\exp(x_{m+1})}^{+\infty} v_\alpha(y_{m+1}, x_{m+1}, t_{m+1}, L_1, L_2) \\ &\quad \cdot f(y_{m+1}, x_{m+1}, t_{m+1} | y_m, x_m)\, dy_{m+1}\, dx_{m+1} \\ &\leqslant e^{-r\Delta t} \end{aligned}$$

and its approximation, by using (14), reads

$$\begin{aligned} &\hat{v}_\alpha(y_m, x_m, t_m, L_1, L_2) \\ &= e^{-r\Delta t} \sum_{k=0}^{N_1-1\,\prime} \sum_{j=0}^{N_2-1\,\prime} Re\left(\phi\left(\frac{k\pi}{b_1-a_1}; \Delta t\right) \exp\left(ik\pi \frac{x_m - a_1}{b_1 - a_1}\right)\right) \\ &\quad \cdot Re\left(\sum_{n=1}^{n_q+2} \frac{2}{b_2 - \exp(\delta_n)} w_n \exp\left(i \frac{j\pi}{b_2 - \exp(\delta_n)} y_m\right) \cos\left(k\pi \frac{\delta_n - a_1}{b_1 - a_1}\right) \int_{L_1}^{L_2} \cos\left(j\pi \frac{y_{m+1} - \exp(\delta_n)}{b_2 - \exp(\delta_n)}\right)\right). \end{aligned} \tag{49}$$

Inserting (47) into (48), then using (45) and (49), gives us

$$\begin{aligned} \left|\epsilon\left(\hat{c}(y_m, x_m, t_m)\right)\right| \leqslant\ & O\left(N_2\left(\epsilon_T\left(X_{m+1}; [a_1, b_1]\right) + \epsilon_F(X_{m+1}; N_1)\right)\right. \\ & \left. + \epsilon_T(Y_{m+1}; b_2) + \epsilon_F(Y_{m+1}; N_2) + N_1 N_2 \epsilon_q(n_q)\right) \\ & + P(N_1, N_2, n_q)\left(\exp\left(-(N_1-1)\nu_1\right) + \exp\left(-(N_2-1)\nu_2\right)\right. \\ & \left. + \exp\left(-(n_q-1)\nu_q\right)\right) \hat{v}_\alpha\left(y_m, x_m, t_m, \exp(\delta_n), y^*_{m+1,n}\right) \\ & + \max_n \left|g(\zeta_n, t_{m+1}) - \hat{c}(\zeta_n, \delta_n, t_{m+1})\right| \cdot \hat{v}_\alpha\left(y_m, x_m, t_m, y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n}\right), \end{aligned} \tag{50}$$

with $\zeta_n \in (y^*_{m+1,n}, y^*_{m+1,n} + \epsilon^*_{m+1,n})$.

For a call option, we have, $\forall n$,

$$\begin{aligned} \left|g(\zeta_n, t_{m+1}) - \hat{c}(\zeta_n, \delta_n, t_{m+1})\right| &= \hat{c}(\zeta_n, \delta_n, t_{m+1}) - g(\zeta_n, t_{m+1}) \\ &\leqslant \hat{c}\left(y^*_{m+1,n}, \delta_n, t_{m+1}\right) - g\left(y^*_{m+1,n}, t_{m+1}\right) \\ &= \hat{c}\left(y^*_{m+1,n}, \delta_n, t_{m+1}\right) - c\left(y^*_{m+1,n}, \delta_n, t_{m+1}\right) \\ &= \epsilon\left(\hat{c}\left(y^*_{m+1,n}, \delta_n, t_{m+1}\right)\right). \end{aligned} \tag{51}$$

Moreover, the value of $\hat{v}_\alpha$ can bounded, as

$$\begin{aligned} \hat{v}_\alpha(y_m, x_m, t_m, L_1, L_2) &\leqslant v_\alpha(y_m, x_m, t_m, L_1, L_2) + \left|\epsilon\left(v_\alpha(y_m, x_m, t_m, L_1, L_2)\right)\right| \\ &= e^{-r\Delta t} + O\left(N_2\left(\epsilon_T\left(X_{m+1}; [a_1, b_1]\right) + \epsilon_F(X_{m+1}; N_1)\right)\right. \\ &\quad \left. + \epsilon_T(Y_{m+1}; b_2) + \epsilon_F(Y_{m+1}; N_2) + N_1 N_2 \epsilon_q(n_q)\right), \end{aligned} \tag{52}$$

where the last step is based on the fact that $\epsilon(v_\alpha(y_m, x_m, t_m, L_1, L_2))$ is the error from approximation (14).

Finally, by using (51) and (52) in (50), and inserting (45), (35) and (36), we reach the conclusion that if $[a_1, b_1]$ and b_2 are carefully chosen, then the truncation errors $\epsilon_T(X_{m+1}; [a_1, b_1])$ and $\epsilon_T(Y_{m+1}; b_2)$ will not be the dominant parts of error (48), and we obtain

$$\begin{aligned} \left|\epsilon\left(\hat{c}(y_m, x_m, t_m)\right)\right| \leqslant\ & \bar{P}(N_1, N_2, n_q)\left(\exp\left(-(N_1-1)\nu_1\right)\right. \\ & \left. + \exp\left(-(N_2-1)\nu_2\right) + \exp\left(-(n_q-1)\nu_q\right)\right), \end{aligned} \tag{53}$$

where $\bar{P}(N_1, N_2, n_q)$ is a term which varies less than exponentially with respect to N_1, N_2, n_q. This concludes the proof. □

In the case of put options or negative-valued errors in the early-exercise points, the same error expression as in (53) can be derived, by a very similar analysis.

Author's personal copy

4. Numerical results

In this section we perform experiments with two different exponential Lévy processes, the Black–Scholes (BS) and the Normal Inverse Gaussian (NIG) processes. In the BS model, the log-asset follows a Brownian motion, and the NIG process [2] is a variance-mean mixture of a Gaussian distribution with an inverse Gaussian. The characteristic function of increment under the NIG model reads:

$$\varphi_{\text{NIG}}(u,t) = \exp\left(iu\mu t - \frac{1}{2}u^2\sigma^2 t\right) \cdot \exp\left(t\delta\left(\sqrt{\alpha^2-\beta^2} - \sqrt{\alpha^2-(\beta+iu)^2}\right)\right),$$

with $\alpha, \delta > 0$ and $\beta \in (-\alpha, \alpha - 1)$. The α-parameter controls the steepness of the density, β is a skewness parameter, and δ is a scale parameter in the sense that the re-scaled parameters $\alpha \to \alpha\delta$ and $\beta \to \beta\delta$ are invariant under location-scale changes of the underlying asset.

The model parameters, used in [11] and [16] for pricing European-style Asian options, are used in Sections 4.1 and 4.2:

- *BS*: $r = 0.0367$, $\sigma = 0.178$;
- *NIG*: $r = 0.0367$, $\sigma = 0.178$, $\alpha = 6.188$, $\beta = -3.894$, $\delta = 0.1622$.

Furthermore the model parameter sets used in [1] are used in Section 4.3.

Reference values used in Sections 4.1 and 4.2 are derived by the ASCOS method *on the GPU*, with $N_1 = N_2 = (n_q/2) + 1 = 4096$, and results from [1] are used as reference values in Section 4.3.

Two types of processors, a CPU (Central Processing Unit), and a GPU (Graphics Processing Unit) with double precision are used and compared to obtain the numerical ASCOS results. On the CPU, an Intel(R) Core(TM)2 Duo CPU E6550 (@ 2.33 GHz Cache size 4 MB), the algorithm is implemented in MATLAB 7.7.0. On the GPU, a Tesla C2070 GPU with 6 GB memory, we coded in Compute Unified Device Architecture (CUDA) [13]. Computing time is recorded in seconds.

4.1. Exponential convergence

Error convergence of early-exercise arithmetic Asian options under the BS model, with 10 and 50 early-exercise dates, using the 2D ASCOS method, are presented in Fig. 1. The horizontal axis presents index d, where in Fig. 1(a), $N_1 = N_2 = 32d$, $(n_q/2) + 1 = 256$, and in Fig. 1(b), we use $N_1 = N_2 = 64d$, $(n_q/2) + 1 = 512$. The vertical axis shows the logarithm of the absolute error. For $\mathcal{M} = 10$ as well as $\mathcal{M} = 50$ an exponential convergence is observed: When N_1 and N_2 increase linearly, the logarithm of the error in the option price decreases accordingly.

When comparing the two plots in Fig. 1, we see that with an increasing number of early-exercise dates we require larger values for N_1, N_2 and n_q to reach the same level of accuracy. With smaller time steps, Δt, the conditional density function between consecutive time steps tends to be peaked, and an accurate approximation by means of cosine expansions then requires an increasing number of terms. The need for a larger value of n_q comes from the fact that the error of the Clenshaw–Curtis quadrature is observed in each term of (14) and there are $N_1 N_2$-terms in total. Therefore, larger values for N_1 and N_2 give rise to a larger n_q-value to ensure the accuracy.

4.2. Computational time and acceleration on the GPU

In this section we present the computational time for our pricing method to reach different accuracy levels on the CPU and GPU. We use the NIG model for the underlying process, while the ASCOS method behaves in a similar way for other exponential Lévy process.

A GPU is an SIMT (Single Instruction, Multiple Threads) machine. In other words, the same command can be executed simultaneously for each data element on each thread of the GPU. Therefore, GPU processing is advantageous for problems that can be expressed in the form of data-parallel computations.

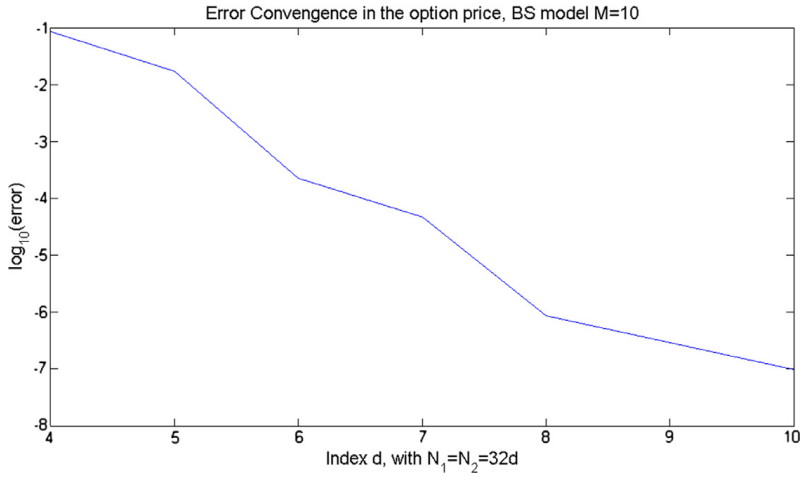
In the early-exercise ASCOS algorithm we process from time step to time step sequentially, however, there are parts of the algorithms for which parallelization is possible. For instance, each early-exercise point, $y^*_{m,n}$, can be independently calculated. The Fourier coefficients, that are represented by a matrix in our pricing method, are computed simultaneously on the GPU at each time step.

We need to perform matrix-vector multiplications, where the summation in each row must be done sequentially. This can be accelerated by the use of shared-memory within each block, which significantly reduces the data-communication time on the GPU.

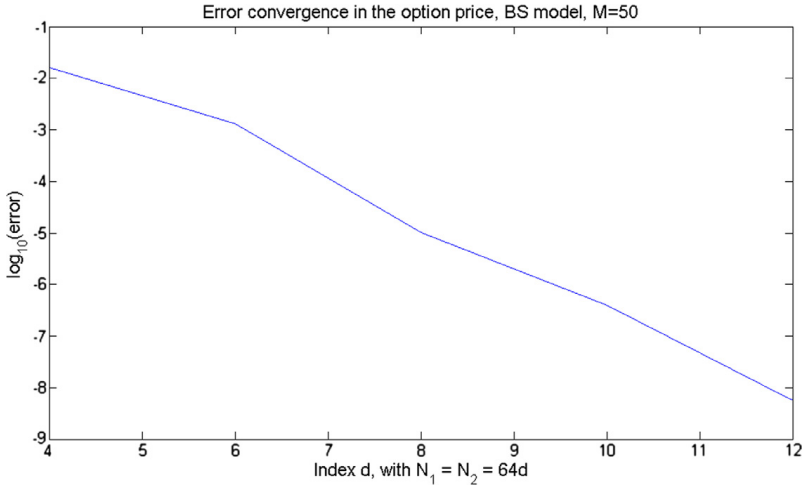
Data transfer between the GPU and the CPU is the bottleneck for most GPU implementations. However, we implemented it in such a way that, independent of the size of the problem, only one number needs to be transferred between the CPU and the GPU, which is the option price which we transfer to the CPU at the end of the computation. When the size of the problem increases, there will be no extra costs due to data transfer.

Tables 1 and 2 present the convergence behavior and computing times for the NIG model with $\mathcal{M} = 10, 50$, on the CPU and GPU, respectively. The error of the option price computed by our method can become arbitrarily small. With

Author's personal copy



(a) $\mathcal{M} = 10$



(b) $\mathcal{M} = 50$

Fig. 1. 2D ASCOS error convergence for early-exercise arithmetic Asian options with different numbers of early-exercise dates, BS model, $S_0 = 100$, $K = 110$.

Table 1
Convergence and computation time of early-exercise arithmetic Asian put options, under the NIG model, with $\mathcal{M} = 10$, $S_0 = 100$ (time in seconds).

2D method			
$N_1 = N_2 = (n_q/2) + 1$	256	384	512
abs. error	1.42e−04	3.14e−07	2.21e−09
CPU time	3560	13 200	32 300
GPU time	4.8	9.1	31.3
Ratio CPU/GPU	740	1450	1030

Table 2
Convergence and computation time of early-exercise arithmetic Asian put options, under the NIG model with $\mathcal{M} = 50$, $S_0 = 100$ (time in seconds).

2D method	$N_1, N_2 = 256$	$N_1, N_2 = 512$	$N_1, N_2 = 768$
abs. error	3.88e−03	1.08e−04	4.90e−07
CPU time	20 300	179 000	395 000
GPU time	20.0	160.7	399.5
Ratio CPU/GPU	1020	1110	990

$N_1 = N_2 = (n_q/2) + 1 = 256$, the method reaches basis point accuracy for $M \leqslant 10$ and when $N_1 = N_2 = (n_q/2) + 1 = 512$, the method reaches basis point accuracy for $M \leqslant 50$.

If a pricing method converges exponentially then the ratio

$$\frac{\log(\text{error}(2^{d+1}))}{\log(\text{error}(2^{d}))}$$

Author's personal copy

Table 3
Comparison between the reference value and convergence speed of our method and method from [1].

	$T = 0.5$, $\sigma = 0.25$	[1] $T = 0.25$, $\sigma = 0.15$
Option value (both methods)	5.33200	2.32084
Grid point [1]	2400	2400
CPU time [1]	2917	2917
Grid point ASCOS	$N_1 = N_2 = n_q/2 + 1 = 256$	$N_1 = N_2 = n_q/2 + 1 = 256$
CPU time ASCOS	3539	3652

should be around 2. When comparing the error with $N_1 = N_2 = (n_q/2) + 1 = 256$ and $N_1 = N_2 = (n_q/2) + 1 = 512$ we find the ratios to be 2.25 for $\mathcal{M} = 10$ and 1.65 for $\mathcal{M} = 50$, which further confirms exponential error convergence. As $\mathcal{M}$ increases the convergence becomes a bit slower. This is due to the fact that the density function of consecutive increments for a Lévy process tend to be peaked.

The GPU computation (on Tesla C2070) is on average 1070 times faster than the CPU computation (in MATLAB) in Table 1 (with $\mathcal{M} = 10$) and on average 1040 times faster in Table 2 (with $\mathcal{M} = 50$).

4.3. Comparison of option values

In this subsection, we compare the early-exercise Asian option prices under the BS model by our ASCOS method with the results in [1]. Results with $\mathcal{M} = 13$ are compared for different parameter sets. Results are shown in Table 3, where we record the number of grid points needed to reach basis point precision and the CPU time involved. Model parameters used in this comparison are $S_0 = 100$, $K = 100$, $r = 0.05$.

Comparing Table 3 with Table 2 we see that the option price for BS model converges slightly faster than for the NIG model. The CPU times are not comparable since they refer to different computers. However, the CPU time recorded in [1] for American option is less than the CPU time for the European counterpart with the same number of grid points which is counterintuitive.

5. Conclusions

In this article, we have developed a robust and efficient pricing method for Asian options with early-exercise features for arithmetic averages, based on a two-dimensional risk-neutral formula. The method is based on Fourier cosine expansions and Clenshaw–Curtis quadrature, and, depending on the smoothness of the density function, may give rise to exponential error convergence. The convergence behavior of the 2D ASCOS method is supported by a detailed error analysis, as well as by various numerical experiments. The flexibility and robustness of the 2D pricing method for different Lévy models and different numbers of early-exercise dates are shown in the numerical experiments. In particular, the Graphics Processing Unit, which supports parallel computing, turns out to be very efficient for the computation of arithmetic Asian option values. The speedup on the GPU is high (just above 1000) as there are many “parallel” computations and almost no data transfer is required.

References

[1] H.B. Ameur, M. Breton, P. L'Ecuyer, A dynamic programming procedure for pricing American-style Asian options, Manag. Sci. 48 (5) (2002) 625–643.
[2] O.E. Barndorff-Nielsen, Normal inverse Gaussian distributions and stochastic volatility modeling, Scand. J. Stat. 24 (1) (1997) 1–13.
[3] E. Benhamou, Fast Fourier Transform for discrete Asian options, J. Comput. Finance 6 (2002) 49–61.
[4] A. Bermúdez, M.R. Nogueiras, C. Vázquez, Numerical solution of variational inequalities for pricing Asian options by high order Lagrange–Galerkin methods, Appl. Numer. Math. 56 (2006) 1256–1270.
[5] J.P. Boyd, Chebychev and Fourier Spectral Methods, 2nd ed., Dover, New York, 2001.
[6] A. Carverhill, L. Clewlow, Flexible convolution, in: From Black Scholes to Black Holes, 1992, pp. 165–171.
[7] T. Dai, Y. Lyuu, Accurate and efficient lattice algorithms for American-style Asian options with range bounds, Appl. Math. Comput. 209 (2) (2009) 238–253.
[8] Y. D'Halluin, P.A. Forsyth, G. Labahn, A semi-Lagrangian approach for American Asian options under jump diffusion, SIAM J. Sci. Comput. 27 (2005) 315–345.
[9] F. Fang, C.W. Oosterlee, A novel option pricing method based on Fourier cosine series expansions, SIAM J. Sci. Comput. 31 (2) (2008) 826–848.
[10] F. Fang, C.W. Oosterlee, Pricing early-exercise and discrete barrier options by Fourier cosine series expansions, Numer. Math. 114 (2009) 27–62.
[11] G. Fusai, A. Meucci, Pricing discretely monitored Asian options under Lévy processes, J. Bank. Finance 32 (2008) 2076–2088.
[12] D. Lemmens, L.Z.J. Liang, J. Tempere, A. De Schepper, Pricing bounds for discrete arithmetic Asian options under Lévy models, Physica A: Stat. Mech. Appl. 389 (22) (2010) 5193–5207.
[13] NVIDIA CUDA, Programming guide, version 4.0, 2011.
[14] L.N. Trefethen, Is Gauss quadrature better than Clenshaw–Curtis, SIAM Rev. 50 (1) (2008) 67–87.
[15] J. Waldvogel, Fast construction of the Fejér and Clenshaw–Curtis quadrature rules, BIT Numer. Math. 46 (2006) 195–202.
[16] B. Zhang, C.W. Oosterlee, Efficient pricing of European-style Asian options under exponential Lévy processes based on Fourier cosine expansions, SIAM J. Financ. Math. 4 (1) (2013) 399–426.