Pricing Asian Option by the FFT with Higher-Order Error Convergence Rate under Lévy Processes

Chun-Yuan Chiu* Tian-Shyr Dai† Yuh-Dauh Lyuu‡

Abstract

Pricing Asian options is a long standing hard problem since there is no analytical formula for the probability density of its payoff even when the process of the underlying asset follows the simple lognormal diffusion process. It is known that the option payoff can be expressed as a recursive function of sums of independent random variables. As a result, the density function of the option payoff can be efficiently approximated by the Fast Fourier Transform (FFT). The advantage of this approach is that we can evaluate Asian options under the more general Lévy process than the lognormal diffusion process. This paper shows that the pricing error of this approach can be decomposed into the truncation error, the integration error, and the interpolation error. We also prove that the pricing results generated by previous algorithms that follow the FFT approach converge quadratically. To improve the error convergence rate, our algorithm reduces the integration error by the higher-order Newton-Cotes formulas and new integration rules derived from the Lagrange interpolating polynomial. The interpolation error are also reduced by the higher-order Newton divided-difference interpolation formula. As a result, our algorithm can be sped up by the FFT to achieve the same time complexity as previous algorithms, but with a faster convergence rate. Numerical results are given to verify the efficiency and the fast convergence of our algorithm.

Keywords: pricing, Fast Fourier Transform, Asian option, Newton-Cotes integration formula

1 Introduction

Options are financial derivatives that give their buyers the right, but not the obligation, to buy or sell the underlying assets for a contractual price, called the exercise price, at a certain date, called the maturity date. They are essential to speculation and the management of financial risk. With the rapid growth and the deregulation of financial markets, many complex options have been structured to meet specific financial goals.

Although an option must have a unique fair price, calculating that price may be computationally difficult. The Asian option suggested by Ingersoll (1987) is an example. The payoff of an Asian option depends on the average price of the underlying asset from the option's starting date to the maturity date. It is hence useful for hedging transactions whose cost depends on the average price of the underlying asset (such as crude oil). Its price is also less subject to price

*Corresponding author. Institute of Information Management, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 300, ROC. E-mail: r94922072@ntu.edu.tw.

†Department of Information and Finance Management, Institute of Information Management and Finance, National Chiao-Tung University, 1001 Ta Hsueh Road, Hsinchu, Taiwan 300, ROC. E-mail: d88006@csie.ntu.edu.tw. The author was supported in part by NSC grant 94-2213-E-033-024 and NCTU research grant for financial engineering and risk management project.

‡Department of Finance and Department of Computer Science & Information Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Road, Taipei, Taiwan 106. E-mail: lyuu@csie.ntu.edu.tw. The author was supported in part by NSC grant 100-2221-E-002-111-MY3.

manipulation. However, it is hard to price Asian options because the probability distribution of the average price of the underlying asset does not have a simple analytical expression. (From now on, the underlying asset is assumed to be stock for convenience.)

By assuming that the stock price follows a lognormal diffusion process, the density of the average stock price can be approximated in various ways: the geometric average of the stock price (Kemna and Vorst, 1990), the lognormal density function (Levy, 1992), the Edgeworth series expansion (Turnbull and Wakeman, 1991), the Taylor expansion (Zhang, 1998), and the reciprocal Gamma distribution (Milevsky and Posner, 1998). In addition, Asian options can also be priced by the lattice method or Monte Carlo simulation (Aingworth et al., 2000; Boyle et al., 1997; Broadie et al., 1999). Empirical studies show that the distribution of stock returns has heavier tails and higher peaks than the lognormal diffusion process (Hosking et al., 2000; Huisman et al., 1998), but most pricing methods rely highly on the lognormal assumption and are hard to extend to other processes. Although the Monte Carlo method is a flexible alternative, it is inefficient and the result is probabilistic.

Carverhill and Clewlow (1990) suggest a pricing algorithm that allows the stock price to follow a wide class of distributions. They show that the average stock price can be obtained by a recursive formula that involves sums of independent random variables. Thus, the density function of the average stock price can be computed by repeated integral convolutions. Specifically, assume that the stock price is sampled at times $t_0, t_1, t_2, \ldots$. The density function of the average stock price from time t_0 to time t_i can be derived from an integral convolution with the density function of the average stock price from time t_0 to time t_{i-1} and the stock price transition probability from time t_{i-1} to time t_i, assuming the density function of the average stock price up to time t_{i-1} and the transition probability are independent. This independence condition can be satisfied if the stock return process follows Lévy processes. Since the density function of the average stock price can not be expressed analytically, it is approximated by keeping the density values at equal-distant grid points in a list by the Carverhill-Clewlow algorithm. Integral convolution is approximated by discrete convolution, which can be efficiently calculated by the Fast Fourier Transform (FFT). As the support of the density function being approximated is infinite, only density values at $2m + 1$ grid points separated at equal distance h within a closed interval with length $2mh$ are kept. This interval is called window below. Interpolations are also used in the Carverhill-Clewlow algorithm. A function of a random variable, say X, is another random variable, say Y. Interpolations are used when estimating the density values of Y with density values of X at grid points.

Benhamou (2002) improves the Carverhill-Clewlow algorithm by significantly narrowing the length of the window. In the Carverhill-Clewlow algorithm, windows should be large enough to contain the bulk of the probability mass of all density functions. Since the bulk of the probability mass of each density function involved in the Carverhill-Clewlow algorithm may be located in different intervals, the length of the window will be large enough to cover all these intervals. As a result, the number of grid points, $2m + 1$, and hence the computational time must be large to keep h and the pricing error small. In Benhamou's algorithm, all random variables involved are shifted so the intervals that contain the bulk of the probability mass of these variables are roughly identical. This paper shows that the time complexity of Benhamou's algorithm is asymptotically identical to the Carverhill-Clewlow algorithm and the error convergence rates of both are identical.

The first major contribution of this paper is to prove that the Carverhill-Clewlow algorithm and the Benhamou algorithm run in $O(nm \ln m)$ time with the error convergence rate $O(h^2)$, where n denotes the number of stock price samples used in the average stock price. In both algorithms, the integral convolutions to evaluate the density values at grid points are approximated by discrete convolutions, which can be calculated by the FFT as they can be rephrased as a multiplication of a Toeplitz matrix and a vector. This multiplication implies that the integral convolution is approximated by the midpoint integration rule, which produce some errors. From

now on, this kind of error is called the integration error. Besides the integration error, there are two other sources of pricing error — the interpolation error and the truncation error. As mentioned before, interpolations are used in both algorithms. This results in the interpolation error. Its error convergence rate depends on the interpolation scheme. For example, it is $O(h^2)$ with the linear interpolation (Isaacson and Keller, 1994). Finally, the truncation error denotes the pricing error introduced by replacing the infinite support of the density function with a finite one. It has been discussed by Černỳ and Kyriakou (2011), who modify the Carverhill-Clewlow algorithm from a forward-induction algorithm to the backward-induction one and then analyze the resulting truncation error. Our numerical results suggest that pricing errors of the Carverhill-Clewlow and Benhamou algorithms are mainly due to the interpolation error and the integration error. Roughly speaking, the quadratic error convergence rates of the Carverhill-Clewlow and Benhamou algorithms are due to the quadratic error convergence rates of the midpoint rule and the linear interpolation. The truncation error of the Carverhill-Clewlow and Benhamou algorithms can be ignored if an appropriate window is used, which does not need to be very large for Benhamou's algorithm.

The second contribution of this paper is a modified Benhamou algorithm. First we try to reduce the integration error by applying a higher-order Newton-Cotes integration formula. However, it is nontrivial to do so due to constraints on the number of sample points for using a higher-order Newton-Cotes integration formula to approximate a definite integral. For example, Simpson's rule and Boole's rule approximate an integral with $2\kappa + 1$ and $4\kappa + 1$ sample points, respectively, where κ is an integer. To bypass the constraints, we derive new integral rules with higher-order error convergence rates by taking advantage of the Lagrange interpolating polynomial and then combine the Newton-Cotes formula with these integral rules. We show that the integral convolution can be discretely approximated by a multiplication of a Toeplitz matrix and a vector plus a multiplication of a sparse matrix and a vector. The former multiplication can be sped up by the FFT, whereas the latter one can be easily computed due to the sparsity of the matrix. Consequently, the resulting algorithm has a better error convergence rate without increasing the time complexity. Fusai and Meucci (2008) show that Carverhill-Clewlow-type algorithms can be used to price discrete Asian options under Lévy processes, such as jump-diffusion processes, double exponential processes, and so on. Our algorithm has the same flexibility. Numerical results are given to verify our claims.

This paper is organized as follows. The financial and mathematical background knowledge is described in section 2. The Carverhill-Clewlow algorithm and the Benhamou algorithm are also introduced in this section. The quadratic error convergence rates and the time complexities of both algorithms are analyzed in section 3. Section 4 describes how the higher-order Newton-Cotes formulas and the higher-order Newton divided-difference interpolation formulas are used to achieve higher-order error convergence rates without increasing the time complexity. Numerical results provided in section 5 verify the superiority of our approach. Section 6 concludes this paper.

2 Preliminaries

2.1 Financial Background Knowledge

Let an Asian call option with strike price K initiate at time 0 and mature at time T. Its underlying stock price at time t is denoted by S_t. The payoff of an Asian call option depends on the average of its underlying stock prices sampled at times $t_0, t_1, t_2, \ldots, t_n$, where $0 = t_0 < t_1 < t_2 < \cdots < t_n = T$ form a partition of the time interval $[0, T]$. For convenience, we define the average stock price as

$$A \equiv \frac{1}{n+1} \sum_{i=0}^{n} S_{t_i}.$$

The value of an Asian call option can be expressed as the expected value of the discounted payoff (Harrison and Pliska, 1981):

$$C = \mathrm{E}\left[e^{-rT}(A-K)^+\right], \tag{1}$$

where r denotes the risk-free interest rate, and $(x)^+$ means $\max(x, 0)$.

The stock return from time t_{i-1} to time t_i, denoted as $R_i \equiv \ln(S_{t_i}/S_{t_{i-1}})$, is assumed to follow a Lévy process. Lévy processes are continuous stochastic processes with stationary independent increments. Specifically, a Lévy process $\{X_t\}_{t\geq 0}$ should satisfy the following conditions:

1. $X_0 = 0$;
2. $X_t - X_s$ and $X_u - X_t$ are independent for $0 \leq s < t < u < \infty$;
3. for all $0 \leq s < t$, $X_t - X_s$ and X_{t-s} have the same distribution;
4. X_t is right continuous almost surely, and $\lim_{t\to 0^-} X_t$ exist for all t.

Most works assume the return process follows a special case of the Lévy process, the Brownian motion:

$$\ln(S_t/S_0) = \left(r - \sigma^2/2\right)t + \sigma W_t,$$

where σ is the volatility of the stock price, and W_t is a standard Brownian motion. Thus the stock price process follows a lognormal diffusion process (LGNO) $S_t = S_0 \exp\{\left(r - \sigma^2/2\right)t + \sigma W_t\}$. Under this assumption, R_i follows a normal distribution with mean $(r - \sigma^2/2)\Delta t_i$ and variance $\sigma^2 \Delta t_i$, where $\Delta t_i \equiv t_i - t_{i-1}$. Though the LGNO model is simple and widely used in the literature, it fails to capture empirical distributions of stock returns, such as high peak, asymmetry and heavy tails. Thus, more general Lévy processes, like the jump-diffusion model (JD), the double exponential model (DE), and the normal inverse Gaussian model (NIG), have been proposed to model the evolution of stock price process (Fusai and Meucci, 2008). Indeed, under many Lévy processes, the density functions of R_i do not have closed-form formulas and are approximated by applying the discrete inverse Fourier transform. The characteristic functions of R_i under the aforementioned models are given in Table 1 (Fusai and Meucci, 2008).

Model	Parameters	Characteristic functions $E(e^{\mathbf{i}uR_i})$
LGNO	r, σ	$\exp\left\{\mathbf{i}u\left(r - \frac{\sigma^2}{2}\right)\Delta t_i - \frac{\sigma^2}{2}u^2\Delta t_i\right\}$
JD	$r, \sigma, \lambda, \alpha, \beta$	$\exp\left\{\mathbf{i}u\left(r - \lambda\left(e^{\alpha+\frac{\beta^2}{2}} - 1\right) - \frac{\sigma^2}{2}\right)\Delta t_i - \frac{\sigma^2 u^2}{2}\Delta t_i + \lambda\Delta t_i\left(e^{i\alpha u - \frac{1}{2}\beta^2 u^2} - 1\right)\right\}$
DE	$r, \sigma, \lambda, \eta_1, \eta_2, p$	$\exp\left\{\mathbf{i}u\left(r - \lambda\left(\frac{(1-p)\eta_2}{\eta_2+1} + \frac{p\eta_1}{\eta_1-1} - 1\right) - \frac{\sigma^2}{2}\right)\Delta t_i - \frac{\sigma^2 u^2}{2}\Delta t_i + \lambda\Delta t_i\left(\frac{(1-p)\eta_2}{\eta_2+\mathbf{i}u} + \frac{p\eta_1}{\eta_1-\mathbf{i}u} - 1\right)\right\}$
NIG	r, δ, α, β	$\exp\left\{\mathbf{i}u\left(r + \delta\left(\sqrt{\alpha^2-(\beta+1)^2} - \sqrt{\alpha^2-\beta^2}\right)\right)\Delta t_i - \delta\Delta t_i\left(\sqrt{\alpha^2-(\beta+\mathbf{i}u)^2} - \sqrt{\alpha^2-\beta^2}\right)\right\}$

Table 1: The characteristic functions of the stock return under different Lévy models.

2.2 Pricing Algorithms Using FFT

Define $X_i \equiv e^{R_i} = S_{t_i}/S_{t_{i-1}}$. Carverhill and Clewlow (1990) express the average stock price A as follows :

$$\begin{aligned}
A &= \frac{1}{n+1}\sum_{i=0}^{n} S_{t_i} = \frac{S_{t_0}}{n+1}\left(1 + X_1 + X_1X_2 + \cdots + X_1X_2\cdots X_n\right) \\
&= \frac{S_{t_0}}{n+1}\left(1 + X_1\left(1 + X_2\left(\cdots X_{n-1}\left(1 + X_n\right)\right)\right)\right) \\
&= \frac{S_{t_0}}{n+1}\left(1 + e^{R_1 + \ln(1+\exp(\cdots + \ln(1+\exp(R_n))))}\right) \\
&= \frac{S_{t_0}}{n+1}\left(1 + e^{B_0}\right), \qquad (2)
\end{aligned}$$

where the recursive sequence $\{B_i\}$ is defined as follows:

$$\begin{aligned} B_{n-1} &= R_n \\ B_{i-1} &= R_i + \ln(1+\exp(B_i)) \qquad i = n-1, n-2, \ldots, 1. \end{aligned} \tag{3}$$

To simplify the notation, we define

$$Y_i \equiv \ln(1+\exp(B_i)); \tag{4}$$

therefore $B_{i-1} = R_i + Y_i$.

Let f_X represent the density function of random variable X for convenience. Then the density function $f_{B_{i-1}}$ can be evaluated as the convolution of f_{R_i} and f_{Y_i}. Thus $f_{B_{i-1}}$ can be efficiently calculated by applying the inverse Fourier transform on the product of the Fourier transforms of f_{R_i} and f_{Y_i}. Since the densities functions $\{f_{B_i}\}$ and $\{f_{Y_i}\}$ can not be solved analytically, Carverhill and Clewlow estimate these probability densities sequences with discrete (inverse) Fourier transform. Specifically, they choose the window to be $[-c, c]$, which contains the bulk of the probability mass of all densities involved in their algorithm. Then $2m+1$ grid points $-c = x_{-m} < \cdots < x_0 = 0 < \cdots < x_m = c$ are selected in the window with equal distance $h = c/m$. Let $\mathbb{B}_{i,k}$ be the estimation for $f_{B_i}(x_k)$ for $k \in [-m, m]$, and equal 0 otherwise. The density function f_{B_i} is approximated by the list $\mathbb{B}_i \equiv (\mathbb{B}_{i,-m}, \ldots, \mathbb{B}_{i,0}, \ldots, \mathbb{B}_{i,m})'$ that stores the estimated densities at the grid points. In a similar manner, density estimations for A, R_i and D_i are stored in list $\mathbb{A}$, $\mathbb{R}_i$ and $\mathbb{D}_i$. Thus the density value of $f_{B_{i-1}}$ at the grid point x_k, denoted as $f_{B_{i-1}}(x_k)$, can be approximated by $\mathbb{B}_{i-1,k}$, which is calculated by applying the inverse discrete Fourier transform on the the product of the discrete Fourier transforms of $\mathbb{R}_i$ and $\mathbb{Y}_i$. This calculation is equivalent to a discrete convolution with the midpoint rule as follows:

$$\begin{aligned} f_{B_{i-1}}(x_k) &= \int_{-\infty}^{\infty} f_{R_i}(x_k - x) f_{Y_i}(x) dx \approx \int_{-c-\frac{h}{2}}^{c+\frac{h}{2}} f_{R_i}(x_k - x) f_{Y_i}(x) dx \\ &\approx \sum_{j=-m}^{m} \mathbb{R}_{i,k-j} \mathbb{Y}_{i,j} h. \end{aligned} \tag{5}$$

Define $\mathbb{B}_{i-1,k} \equiv \sum_{j=-m}^{m} \mathbb{R}_{i,k-j} \mathbb{Y}_{i,j} h$ for convenience. Although the use of midpoint rule in the Carverhill-Clewlow algorithm is not explicitly mentioned in Carverhill and Clewlow (1990), this property is key for its quadratic error convergence rate, as the paper will show later. The approximation formula (5) can be succinctly expressed as a multiplication,

$$\mathbb{B}_{i-1} = M_i \mathbb{Y}_i h, \tag{6}$$

where M_i is the following matrix,

$$M_i \equiv \begin{pmatrix} \mathbb{R}_{i,0} & \cdots & \mathbb{R}_{i,-m+1} & \mathbb{R}_{i,-m} & 0 & \cdots & 0 \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ \mathbb{R}_{i,m-1} & \cdots & \mathbb{R}_{i,0} & \mathbb{R}_{i,-1} & \mathbb{R}_{i,-2} & \cdots & 0 \\ \mathbb{R}_{i,m} & \cdots & \mathbb{R}_{i,1} & \mathbb{R}_{i,0} & \mathbb{R}_{i,1} & \cdots & \mathbb{R}_{i,-m} \\ 0 & \cdots & \mathbb{R}_{i,2} & \mathbb{R}_{i,1} & \mathbb{R}_{i,0} & \cdots & \mathbb{R}_{i,-m+1} \\ \vdots & & \vdots & \vdots & \vdots & & \vdots \\ 0 & \cdots & 0 & \mathbb{R}_{i,m} & \mathbb{R}_{i,m-1} & \cdots & \mathbb{R}_{i,0} \end{pmatrix}. \tag{7}$$

In practice, M_i is padded with needed zeros due to the finite length of list $\mathbb{R}_i$. Note that M_i is a Toeplitz matrix, that is,

$$M_i(j,k) = M_i(j-1, k-1), \quad \forall j, k = 2, 3, \ldots, n, \tag{8}$$

where $M_i(j,k)$ denotes the element located at the i-th row and the j-th column. Multiplying an $n \times n$ Toeplitz matrix by a $n \times 1$ vector can be efficiently done in time complexity $O(n \log n)$ via the FFT (Cormen et al., 2001) and this property will be used in our fast convergent algorithms.

Recall that f_{R_i} is not analytically available under some Lévy processes. To obtain $\mathbb{R}_i$ for these processes, we apply the discrete inverse Fourier transform on the characteristic function of R_i listed in Table 1, which can be done in $O(m \ln m)$ time by the FFT. In addition, the error convergence rate of this approximation can be enhanced by using the Newton-Cotes formulas, as shown in Appendix A. Finally, we can calculate $\mathbb{B}_0$ with Eq. (3), then substitute $\mathbb{B}_0$ into Eq. (2) to obtain $\mathbb{A}$, and then substitute $\mathbb{A}$ into Eq. (1) to evaluate the option value C.

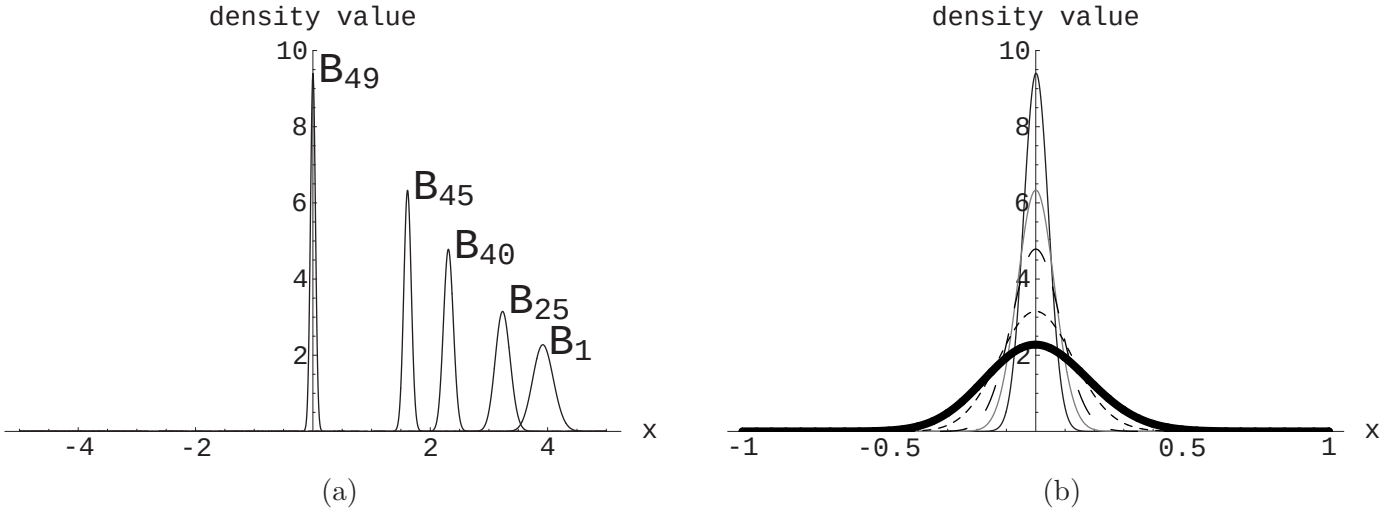
Algorithm 1 Carverhill-Clewlow algorithm.

1: Evaluate the density values in $\mathbb{B}_{n-1}$ by the density function of $f_{B_{n-1}}$.
2: **for** $i = n-1$ down to 1 **do**
3: Calculate the density values of $\mathbb{Y}_i$ by linear interpolation with $\mathbb{B}_i$.
4: Calculate $\mathbb{B}_{i-1}$ defined in Eq. (6) via FFT.
5: **end for**
6: Evaluate the option value C by the density function $\mathbb{B}_0$ and Eqs. (1) and (2).

The Carverhill-Clewlow algorithm is described in Algorithm 1. Note that from the definition of Y_i given by Eq. (4), we have

$$f_{Y_i}(x) = \begin{cases} f_{B_i}(\ln(e^x - 1))\frac{e^x}{e^x - 1} & \text{for all } x > 0 \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$

Thus, in line 3 of the algorithm, $\mathbb{Y}_i$ can be obtained by applying interpolation to $\mathbb{B}_i$ with the help of Eq. (9). Our paper will show that both the Carverhill-Clewlow algorithm and the Benhamou's re-centering algorithm (introduced later) run in $O(nm \ln(m))$ with an error convergence rate $O(h^2)$.



(a) (b)

Figure 1: The Plot Densities before and after Re-centering.
Panel (a) denotes the densities of B_i (approximated by $\mathbb{B}_i$) used in the Carverhill-Clewlow algorithm. Panel (b) denotes the densities of D_i (approximated by $\mathbb{D}_i$) used in the Benhamou's algorithm. The thin black line, the thin gray line, the long dash line, the short dash line, and the thick black line denote the densities of D_{49}, D_{45}, D_{40}, D_{25}, and D_1, respectively. The stock return is assumed to follow a normal distribution with numerical setting: $S_{t_0} = 100$, $r = 0.1$, $\sigma = 0.3$, $T = 1$, $n = 50$.

Benhamou (2002) argues that the bulk of the probability mass of $\mathbb{B}_i$ shifts significantly as i decreases. This point is illustrated in panel (a) of Figure 1. Thus the window size ($= 2c$) must be large enough to keep the bulk of each density function's mass within the window. Therefore, m should be large to keep $h = c/m$ (and hence the pricing error) small. As a result, the time complexity of $O(nm\ln(m))$ may run slowly. To address the problem, Benhamou "re-centered" the density functions of $\{B_i\}$ by introducing a sequence $\{D_i\}$ defined by

$$D_i \equiv B_i - \mu_i$$

, where μ_i approximates $\mathrm{E}[B_i]$ by the following recurrence relation:

$$\begin{aligned} \mu_{n-1} &= E[R_n] \\ \mu_{i-1} &= E[R_i] + \ln(1 + \exp(\mu_i)), \qquad i = n-1, n-2, \ldots, 1. \end{aligned}$$

In consequence, the plot of $\{\mathbb{D}_i\}$, the sequence of lists store the densities estimations for the sequence of probability densities $\{f_{D_i}\}$, would not shift as significantly as the plot of $\{\mathbb{B}_i\}$ as plotted in Figure 1. In addition, they define

$$Z_i \equiv Y_i - \mu_{i-1} = \ln(1 + \exp(D_i + \mu_i)) - \mu_{i-1} \tag{10}$$

to replace the role of Y_i in the Carverhill-Clewlow algorithm; as a result, the recurrence relation (3) can be rewritten as

$$\begin{aligned} D_{n-1} &= R_n - \mu_{n-1} \\ D_{i-1} &= R_i + Z_i \qquad\qquad i = n-1, n-2, \ldots, 1. \end{aligned} \tag{11}$$

Algorithm 2 sums up Benhamou's algorithm, where the density estimations for f_{Z_i} are stored in the list $\mathbb{Z}_i$.

Algorithm 2 Benhamou's algorithm.

1: Evaluate the densities values in the list $\mathbb{D}_{n-1}$.
2: **for** $i = n-1$ down to 1 **do**
3: Calculate $\mathbb{Z}_i$ by linear interpolation with $\mathbb{D}_i$ (see Eq. (10)).
4: Calculate $\mathbb{D}_{i-1}$, the discrete convolution of $\mathbb{R}_i$ and $\mathbb{Z}_i$, via the FFT.
5: **end for**
6: Evaluate the option value C by the density function $\mathbb{D}_0$ and Eqs. (1) and (2).

2.3 Newton-Cotes Integration Formulas

The Newton-Cotes integration formulas approximate an integral based on numerically evaluating the integrand at equal-spaced points, like the grid points $\{x_i\}$ defined above. For example, $\int_a^b g(x)dx$ can be approximated by $\int_a^b \phi(x)dx$, where $\phi(x)$ denotes an interpolation formula that passes through certain points in the set $\{(x_i, g(x_i)\}$. Recall that $h = c/m$ denotes the distance between two adjacent grid points in $\{x_i\}$. For example, the integral $\int_{x_i-\frac{h}{2}}^{x_i+\frac{h}{2}} g(x)dx$ can be approximated by setting $\phi(x)$ as a constant function that passes through $(x_i, g(x_i))$ as follows:

$$\int_{x_i-\frac{h}{2}}^{x_i+\frac{h}{2}} g(x)dx = hg(x_i) + O(h^3).$$

By summing up the above formula with respect to $i = -m, -m+1, \ldots, m$, the midpoint rule obtains:

$$\int_{x_{-m}-\frac{h}{2}}^{x_m+\frac{h}{2}} g(x)dx = h\sum_{i=-m}^{m} g(x_i) + (2m+1)O(h^3) = h\sum_{i=-m}^{m} g(x_i) + O(h^2) \approx h\sum_{i=-m}^{m} g(x_i).$$

Evaluating the convolutions (3) and (11) in Benhamou (2002); Carverhill and Clewlow (1990) implicitly uses the midpoint rule and thus we can prove that the pricing errors of their methods are $O(h^2)$. To improve the error convergence rate, higher-order Newton-Cotes integration formulas can be used. For example, Simpson's rule can be derived by using a quadratic $\phi(x)$ as follows:

$$\int_{x_i}^{x_{i+2}} g(x)dx = \frac{h}{3}(g(x_i) + 4g(x_{i+1}) + g(x_{i+2})) + O(h^5). \tag{12}$$

By summing up for $i = -m, -m+2, \ldots, m-2$, the composite Simpson's rule obtains (Isaacson and Keller, 1994):

$$\int_{x_{-m}}^{x_m} g(x)dx = \frac{h}{3}\left(\sum_{i=-m,-m+2,\ldots,m-2} (g(x_i) + 4g(x_{i+1}) + g(x_{i+2}))\right) + O(h^4). \tag{13}$$

Note that the number of sample points x_{-m}, x_{-m+1}, $\ldots$, x_m must be odd for the composite Simpson's rule. Similarly, approximating $g(x)$ by a quartic interpolating polynomial $\phi(x)$ gives Boole's rule (Isaacson and Keller, 1994):

$$\begin{aligned}\int_{x_{-m}}^{x_m} g(x)dx &= \frac{h}{45}\left(\sum_{i=-m,-m+4,\ldots,m-4} (14g(x_i) + 64g(x_{i+1}) + 24g(x_{i+2})\right. \\ &\quad \left. +64g(x_{i+3}) + 14g(x_{i+4}))\right) + O(h^6).\end{aligned} \tag{14}$$

The number of sample points should be $4\ell + 1$, for some positive integer ℓ for Boole's rule.

2.4 The Hadamard Product

Given two matrices $\mathbb{M}$ and $\mathbb{M}'$ with the same size, the Hadamard product is defined by

$$(\mathbb{M} * \mathbb{M}')(i,j) \equiv \mathbb{M}(i,j)\mathbb{M}'(i,j).$$

Note that the Hadamard product of two Toeplitz matrices (see the definition in Eq. (8)) is again a Toeplitz matrix, and multiplying a Toeplitz matrix by a vector can be efficiently done in time complexity $O(n \log n)$ via the FFT (Cormen et al., 2001). This property will be used to speed up the evaluation of higher-order Newton-Cotes integration formula with the FFT.

3 Analysis of the Carverhill-Clewlow and Benhamou Algorithms

This section will show that the pricing error of the Benhamou algorithm converges at a rate of $O(h^2)$ with time complexity $O(nm \ln m)$. As the Carverhill-Clewlow algorithm is the Benhamou algorithm with $\mu_i \equiv 0$, the same results hold for the Carverhill-Clewlow algorithm.

3.1 Error Convergence Rate

Recall that three different types of errors are introduced by the Benhamou algorithm: the truncation error, the interpolation error and the integration error. The truncation error denotes the error caused by truncating the probability density out of the window $[-c, c]$, which can be neglected when the window is large enough. The interpolation error denotes the error caused by estimating the density values in $\mathbb{Z}_i$ with $\mathbb{D}_i$ by interpolation in line 3 of Algorithm 2. The integration error denotes the error caused by estimating f_{D_i} and f_{R_i} with discrete convolution in line 4 of Algorithm 2. Note that the rate for $\mathbb{R}_i$ to converge to f_{R_i} depends on the kind of Newton-Cotes formula we choose. When the midpoint rule is used, we have

$$|\mathbb{R}_{i,k} - f_{R_i}(x_k)| = O(h^2). \tag{15}$$

We will show that the whole pricing error, contributed by the accumulation of the integration and interpolation errors, converges asymptotically to zero at a rate of $O(h^2)$.

First, we prove the approximated density values of the grid list $\mathbb{D}_i$ converge asymptotically in $O(h^2)$ by induction on i. For the base case ($i = n-1$), $\mathbb{D}_{n-1,k}$ stores the approximated value of $f_{R_{n-1}}(x_k + E(R_n))$ by interpolating $\mathbb{R}_{n-1}$. Since the error in $\mathbb{R}_{n-1}$ (or $|\mathbb{R}_{n-1,k} - f_{R_{n-1}}(x_k)|$) is $O(h^2)$ (see Eq. (15)), the error to obtain $\mathbb{D}_{n-1,k}$ (or $|\mathbb{D}_{n-1,k} - f_{D_{n-1}}(x_k)|$) by interpolating $\mathbb{R}_{n-1}$ is $O(h^2)$ (Isaacson and Keller, 1994). The inductive step will show that $|\mathbb{D}_{i-1,k} - f_{D_{i-1}}(x_k)| = O(h^2)$ given $|\mathbb{D}_{i,k} - f_{D_i}(x_k)| = O(h^2)$. By the triangle inequality,

$$
\begin{aligned}
\left|\mathbb{D}_{i-1,k} - f_{D_{i-1}}(x_k)\right| &= \left|\sum_{j=-m}^{m} \mathbb{Z}_{i,j}\mathbb{R}_{i,k-j}h - f_{D_{i-1}}(x_k)\right| \\
&\le \sum_{j=-m}^{m} \left|\mathbb{Z}_{i,j} - f_{Z_i}(x_j)\right| \mathbb{R}_{i,k-j}h + \left|\sum_{j=-m}^{m} f_{Z_i}(x_j)\mathbb{R}_{i,k-j}h - f_{D_{i-1}}(x_k)\right| \\
&\le \underbrace{\sum_{j=-m}^{m} \left|\mathbb{Z}_{i,j} - f_{Z_i}(x_j)\right| \mathbb{R}_{i,k-j}h}_{\mathbf{A}} + \underbrace{\sum_{j=-m}^{m} f_{Z_i}(x_j) \left|\mathbb{R}_{i,k-j} - f_{R_i}(x_{k-j})\right| h}_{\mathbf{B}} \\
&\quad + \underbrace{\left|\sum_{j=-m}^{m} f_{Z_i}(x_j) f_{R_i}(x_{k-j})h - \int_{x_{-m}-\frac{h}{2}}^{x_m+\frac{h}{2}} f_{Z_i}(x) f_{R_i}(x_k - x)dx\right|}_{\mathbf{C}} \\
&\quad + \underbrace{\left|\int_{x_{-m}-\frac{h}{2}}^{x_m+\frac{h}{2}} f_{Z_i}(x) f_{R_i}(x_k - x)dx - f_{D_{i-1}}(x_k)\right|}_{\mathbf{D}}.
\end{aligned}
$$

In expression **B**, we have $|\mathbb{R}_{i,k-j} - f_{R_i}(x_{k-j})| = O(h^2)$ as mentioned in Eq. (15) and

$$\sum_{j=-m}^{m} f_{Z_i}(x_j)h = 1 + O(h^2), \tag{16}$$

since $\sum_{j=-m}^{m} f_{Z_i}(x_j)h$ is a truncated approximation for $\int_{-\infty}^{\infty} f_{Z_i}(x)dx = 1$ by the midpoint rule. Thus, we obtain $\mathbf{B} = O(h^2)$.

In expression **A**, we have

$$\sum_{j=-m}^{m} f_{R_i}(x_{k-j})h = 1 + O(h^2),$$

by the similar arguments leading to Eq. (16). Therefore,

$$
\begin{aligned}
\sum_{j=-m}^{m} \mathbb{R}_{i,k-j}h &\le \sum_{j=-m}^{m} f_{R_i}(x_{k-j})h + \sum_{j=-m}^{m} \left|\mathbb{R}_{i,k-j} - f_{R_i}(x_{k-j})\right| h \\
&= 1 + O(h^2) + (2m+1)\,O(h^3) = 1 + O(h^2).
\end{aligned} \tag{17}
$$

$\mathbb{Z}_{i,j}$ is obtained by applying linear interpolation on Eq. (10) with the density values from $\mathbb{D}_i$. Due to the premise of the inductive step, $|\mathbb{D}_{i,k} - f_{D_i}(x_k)| = O(h^2)$ holds, and the error to obtain $\mathbb{Z}_{i,j}$, i.e., $|\mathbb{Z}_{i,j} - f_{Z_i}(x_j)|$ by interpolating $\mathbb{D}_i$ is $O(h^2)$ (Isaacson and Keller, 1994). Substituting this and Eq. (17) into expression **A**, we have $\mathbf{A} = O(h^2)(1 + O(h^2)) = O(h^2)$. Block **D** can be ignored since the truncation error can be neglected by taking a large window,

and expression **C** denotes the integration error of the midpoint rule, which is $O(h^2)$. To sum up, $\left|\mathbb{D}_{i-1,k} - f_{D_{i-1}}(x_k)\right| = O(h^2)$. Thus, by induction, $|\mathbb{D}_{0,k} - f_{D_0}(x_k)| = O(h^2)$.

Finally, we proceed to derive the option price $\int_0^\infty f_A(x)(x-K)^+dx$ in terms of f_{D_0} and show that the pricing error of the Benhamou algorithm is $O(h^2)$. The relationship between random variables A and D_0 can be derived by Eq. (2) as follows:

$$A = \frac{S_{t_0}}{n+1}(1 + e^{D_0+\mu_0}).$$

Thus the option value can be rewritten by changing variables as follows:

$$\begin{aligned}\int_0^\infty f_A(y)(y-K)^+dy &= \int_{-\infty}^\infty f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)^+dx \\ &= \int_{\ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0}^\infty f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx \qquad (19)\end{aligned}$$

For convenience, we divide the above integral into the principal term

$$\int_{x_{k_0}-\frac{h}{2}}^\infty f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx, \qquad (20)$$

and the remainder term

$$\int_{\ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0}^{x_{k_0}-\frac{h}{2}} f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx, \qquad (21)$$

where k_0 is the smallest integer such that $x_{k_0} - \frac{h}{2} \geq \ln\left(\frac{K(n+1)}{S_{t_0}} - 1\right) - \mu_0$. The principal term (20) is easy to approximate by applying the midpoint rule with the grid list $\mathbb{D}_0$. The approximation error can be shown to be $O(h^2)$ as follows:

$$\begin{aligned}&\left|\int_{x_{k_0}-\frac{h}{2}}^\infty f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx - h\sum_{k=k_0}^m \mathbb{D}_{0,k}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_k+\mu_0}\right)-K\right)\right| \\ \leq\ &\left|\int_{x_{k_0}-\frac{h}{2}}^\infty f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx - \int_{x_{k_0}-\frac{h}{2}}^{x_m+\frac{h}{2}} f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx\right| \qquad (22) \\ +\ &\left|\int_{x_{k_0}-\frac{h}{2}}^{x_m+\frac{h}{2}} f_{D_0}(x)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x+\mu_0}\right)-K\right)dx - h\sum_{k=k_0}^m f_{D_0}(x_k)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_k+\mu_0}\right)-K\right)\right| \qquad (23) \\ +\ &\left|h\sum_{k=k_0}^m f_{D_0}(x_k)\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_k+\mu_0}\right)-K\right) - h\sum_{k=k_0}^m \mathbb{D}_{0,k}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_k+\mu_0}\right)-K\right)\right|.\end{aligned}$$

Since the truncation error in Eq. (22) can be neglected by picking a large window and the integration error of the midpoint rule (23) is $O(h^2)$, the RHS of the above inequality is at most

$$\begin{aligned}&O(h^2) + h\sum_{k=k_0}^m |f_{D_0}(x_k) - \mathbb{D}_{0,k}|\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_k+\mu_0}\right)-K\right) \\ =\ &O(h^2) + hO(h^2)\sum_{k=k_0}^m \left(\frac{S_{t_0}}{n+1}\left(1+e^{x_k+\mu_0}\right)-K\right) \\ \leq\ &O(h^2) + hO(h^2)m\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_m+\mu_0}\right)-K\right) \\ =\ &O(h^2),\end{aligned}$$

since $m = O(1/h)$. On the other hand, the remainder term (21) is simply an integral of a smooth function over a closed interval with length less than h, and the integrand equals zero at the integral's lower limit, which is bounded by $O(h^2)$ as shown in Appendix B. Thus, the error introduced by neglecting the remainder term is $O(h^2)$. Consequently, the Benhamou algorithm converges quadratically.

In our algorithm, the convergence rates of the interpolation and integration errors will be improved by our versions of higher-order Newton-Cotes formulas and the higher-order Newton divided-difference interpolation formula. The error analysis is similar.

3.2 Analysis of Time Complexity

Note that $\{\mathbb{R}_i\}$ $(1 \leq i \leq n)$ is evaluated by the inverse Fourier transform on the characteristic function of R_i in $O(nm \ln m)$ time. Then $\mathbb{D}_{n-1}$ is evaluated in $O(m)$ time by applying interpolation on $\mathbb{R}_{n-1}$. Next, $\mathbb{D}_0$ is evaluated by alternately repeating interpolations and convolutions $n-1$ times. $\mathbb{Z}_i$ is evaluated by applying interpolation on $\mathbb{D}_i$ and it takes $O(m)$ time. Convoluting $\mathbb{R}_i$ and $\mathbb{Z}_i$ takes $O(m \ln m)$ time. Thus it takes $O((n-1)(m + m \ln m)) = O(nm \ln m)$ time to evaluate $\mathbb{D}_0$. The option value defined in Eq. (1) can be evaluated by numerical integration with $\mathbb{D}_0$ in $O(m)$ time. To sum up, the total time complexity is $O(nm \ln m)$ time.

4 Our Fast Convergent Algorithms

We will first demonstrate how the error convergence rate can be improved from $O(h^2)$ to $O(h^4)$ by suppressing the integration error and the interpolation error to $O(h^4)$ by our modified Simpson's rule and the third-order Newton divided-difference interpolation formula, respectively. The time complexity remains $O(nm \ln m)$. Then we show that the error convergence rate can be further improved by using the higher-order Newton-Cotes formula and the Newton divided-difference interpolation formula.

4.1 Algorithms with Higher-order Convergence Rates

First we try to improve the integration error due to convolution.[1] In the Benhamou algorithm, evaluating the grid list $\mathbb{D}_i$ by convolution can be reduced to matrix multiplication $\mathbb{D}_{i-1} = M_i \mathbb{Z}_i h$, where M_i is a Toeplitz matrix in Eq. (6). Note that this computation can be sped up by the FFT. As shown earlier, $M_i \mathbb{Z}_i h$ can be thought of as applying $2m+1$ midpoint rules. To improve the error convergence rate, we use higher-order Newton-Cotes formulas such as Simpson rule to replace the midpoint rule. However, this idea may encounter two problems. First, some of the $2m+1$ integrations can not be directly approximated by Simpson's rule since there are certain constraints on the number of sample points. Hence modifications on Simpson's rule are needed. Second, the matrix M_i is no longer Toeplitz after the modifications; thus the computation of $M_i \mathbb{Z}_i h$ can no longer be sped up by the FFT. Fortunately, the modified M_i can be decomposed into the sum of a Toeplitz matrix and a sparse matrix. As a result, the time complexity for computing $M_i \mathbb{Z}_i h$ remains unchanged, as we will show later.

Now we describe how to use the adapted Simpson's rule to bypass the constraint on the number of sample points. To estimate $f_{D_{i-1}}(x_k)$ that involves an odd number of sample points with the constraint $x_k < 0$ (i.e., k is a negative even number), Simpson's rule is directly applied

[1] Note that the error due to the discrete inverse Fourier transform can be improved by using the Newton-Cotes formulas.

to obtain

$$\begin{aligned} f_{D_{i-1}}(x_k) &= \int_{-\infty}^{\infty} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &\approx \int_{x_{-m}}^{x_{m+k}} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &\approx h \sum_{j=-m}^{m+k} w_j \mathbb{R}_{i,k-j} \mathbb{Z}_{i,j} \\ &\equiv \mathbb{D}_{i-1,k}. \end{aligned} \tag{24}$$

where the range of integration is $x_{-m} < x < x_{k+m}$ to make both $x_k - x$ and x in Eq. (24) remain in the window, and $\{w_j\}, \{\hat{w}_j\}$ and $\{\tilde{w}_j\}$ denote three sequences of Simpson's coefficients: $\{\frac{1}{3}, \frac{4}{3}, \frac{2}{3}, \frac{4}{3}, \ldots, \frac{4}{3}, \frac{1}{3}\}$. Similarly, if $x_k > 0$ (i.e., k is a positive even number), we have

$$\begin{aligned} f_{D_{i-1}}(x_k) &\approx \int_{x_{k-m}}^{x_m} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &\approx h \sum_{j=k-m}^{m} \tilde{w}_j \mathbb{R}_{i,k-j} \mathbb{Z}_{i,j} \\ &\equiv \mathbb{D}_{i-1,k}. \end{aligned}$$

On the other hand, if an even number of sample points (i.e., k is odd) are involved in the numerical integration, we apply a modified integration rule with a shorter integration interval as follows:

$$\begin{aligned} \int_{x_j}^{x_j+h} g(x) dx &\approx \frac{5g(x_j) + 8g(x_j + h) - g(x_j + 2h)}{12} h \\ &\approx \frac{-g(x_j - h) + 8g(x_j) + 5g(x_j + h)}{12} h. \end{aligned} \tag{25}$$

This integration formula can be derived by integrating the Lagrange interpolating formula of $g(x)$, and this quadrature can be proved to have the same error convergence rate $O(h^4)$ as Simpson's rule, as shown in Appendix B. Numerical integrations with an even number of sample points (i.e., k is odd) can now be solved by combining Simpson's rule and the modified integration rule (25). If k is a negative odd number (i.e., an even number of sample points with $x_k < 0$), we have

$$\begin{aligned} f_{D_{i-1}}(x_k) &\approx \int_{x_{-m}}^{x_{m+k}} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &= \int_{x_{-m}}^{x_{-m+1}} f_{R_i}(x_k - x) f_{Z_i}(x) dx + \int_{x_{-m+1}}^{x_{m+k}} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &\approx \frac{5\mathbb{R}_{i,k+m}\mathbb{Z}_{i,-m} + 8\mathbb{R}_{i,k+m-1}\mathbb{Z}_{i,-m+1} - \mathbb{R}_{i,k+m-2}\mathbb{Z}_{i,-m+2}}{12} h + h \sum_{j=-m+1}^{m+k} \hat{w}_j \mathbb{R}_{i,k-j} \mathbb{Z}_{i,j} \\ &\equiv \mathbb{D}_{i-1,k}, \end{aligned}$$

where we use the modified integration rule (25) and Simpson's rule, and $\hat{w}_j$ denotes Simpson's

coefficients. Similarly, if $x_k > 0$ (i.e., k is a positive odd number), we have

$$\begin{aligned} f_{D_{i-1}}(x_k) &\approx \int_{x_{k-m}}^{x_m} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &= \int_{x_{k-m}}^{x_{m-1}} f_{R_i}(x_k - x) f_{Z_i}(x) dx + \int_{x_{m-1}}^{x_m} f_{R_i}(x_k - x) f_{Z_i}(x) dx \\ &\approx h \sum_{j=k-m}^{m-1} \tilde{w}_j \mathbb{R}_{i,k-j} \mathbb{Z}_{i,j} + \frac{-\mathbb{R}_{i,k-m+2}\mathbb{Z}_{i,m-2} + 8\mathbb{R}_{i,k-m+1}\mathbb{Z}_{i,m-1} + 5\mathbb{R}_{i,k-m}\mathbb{Z}_{i,m}}{12} h \\ &\equiv \mathbb{D}_{i-1,k}. \end{aligned}$$

By combining the above results, the convergence rate of the integration error can be improved by replacing $\mathbb{D}_{i-1} = M_i \mathbb{Z}_i h$ with $\mathbb{D}_i = \left(M^{\text{Simpson}} * M_i\right) \mathbb{Z}_{i+1} h$, where M^{Simpson} is given by

$$\begin{pmatrix} \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{1}{3} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \frac{5}{12} & \frac{8}{12}+\frac{1}{3} & -\frac{1}{12}+\frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{1}{3} & 0 & 0 & 0 & 0 \\ \frac{5}{12} & \frac{8}{12}+\frac{1}{3} & -\frac{1}{12}+\frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{1}{3} & 0 & 0 & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 & 0 \\ \frac{5}{12} & \frac{8}{12}+\frac{1}{3} & -\frac{1}{12}+\frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} & 0 \\ \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} \\ 0 & \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3}-\frac{1}{12} & \frac{1}{3}+\frac{8}{12} & \frac{5}{12} \\ 0 & 0 & \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} \\ 0 & 0 & 0 & \frac{1}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3}-\frac{1}{12} & \frac{1}{3}+\frac{8}{12} & \frac{5}{12} \\ 0 & 0 & 0 & 0 & \frac{1}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & \frac{1}{3} & \frac{4}{3} & \frac{2}{3} & \cdots & \frac{4}{3} & \frac{2}{3} & \frac{4}{3}-\frac{1}{12} & \frac{1}{3}+\frac{8}{12} & \frac{5}{12} \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & \frac{1}{3} & \frac{4}{3} & \cdots & \frac{2}{3} & \frac{4}{3} & \frac{2}{3} & \frac{4}{3} & \frac{1}{3} \end{pmatrix}.$$

Because $M^{\text{Simpson}} * M_i$ is not a Toeplitz matrix, $\mathbb{D}_i = \left(M^{\text{Simpson}} * M_i\right) \mathbb{Z}_{i+1} h$ can not be evaluated by the FFT. To address this problem, we decompose M^{Simpson} into the sum of M^{Toeplitz} and M^{adjust}, where

$$M^{\text{Toeplitz}} \equiv \frac{1}{3} \begin{pmatrix} 2 & 4 & 2 & 4 & 2 & \cdots & 4 & 1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\ 4 & 2 & 4 & 2 & 4 & \cdots & 2 & 4 & 1 & \cdots & 0 & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 2 & 4 & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & \cdots & 1 & 0 & 0 & 0 & 0 \\ 4 & 2 & 4 & 2 & 4 & \cdots & 2 & 4 & 2 & \cdots & 4 & 1 & 0 & 0 & 0 \\ 2 & 4 & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & \cdots & 2 & 4 & 1 & 0 & 0 \\ 4 & 2 & 4 & 2 & 4 & \cdots & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & 1 & 0 \\ 1 & 4 & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & \cdots & 2 & 4 & 2 & 4 & 1 \\ 0 & 1 & 4 & 2 & 4 & \cdots & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & 2 & 4 \\ 0 & 0 & 1 & 4 & 2 & \cdots & 4 & 2 & 4 & \cdots & 2 & 4 & 2 & 4 & 2 \\ 0 & 0 & 0 & 1 & 4 & \cdots & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & 2 & 4 \\ 0 & 0 & 0 & 0 & 1 & \cdots & 4 & 2 & 4 & \cdots & 2 & 4 & 2 & 4 & 2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & 1 & 4 & 2 & \cdots & 4 & 2 & 4 & 2 & 4 \\ 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 1 & 4 & \cdots & 2 & 4 & 2 & 4 & 2 \end{pmatrix}$$

and

$$M^{\text{adjust}} \equiv \begin{pmatrix}
-\frac{1}{3} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
-\frac{11}{12} & \frac{1}{3} & -\frac{1}{12} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
-\frac{1}{3} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
-\frac{11}{12} & \frac{1}{3} & -\frac{1}{12} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
-\frac{1}{3} & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
-\frac{11}{12} & \frac{1}{3} & -\frac{1}{12} & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & -\frac{1}{12} & \frac{1}{3} & -\frac{11}{12} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & -\frac{1}{3} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & -\frac{1}{12} & \frac{1}{3} & -\frac{11}{12} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & -\frac{1}{3} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & -\frac{1}{12} & \frac{1}{3} & -\frac{11}{12} \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & -\frac{1}{3}
\end{pmatrix}.$$

Since the Hadamard product and the standard matrix product both satisfy the distribution law with matrix addition, we have the following identity:

$$\begin{aligned}
\mathbb{D}_i &= \left(M^{\text{Simpson}} * M_i\right) \mathbb{Z}_{i+1} h \\
&= \left(\left(M^{\text{Toeplitz}} + M^{\text{adjust}}\right) * M_i\right) \mathbb{Z}_{i+1} h \\
&= \left(M^{\text{Toeplitz}} * M_i + M^{\text{adjust}} * M_i\right) \mathbb{Z}_{i+1} h \\
&= \left(M^{\text{Toeplitz}} * M_i\right) \mathbb{Z}_{i+1} h + \left(M^{\text{adjust}} * M_i\right) \mathbb{Z}_{i+1} h. \qquad (26)
\end{aligned}$$

Therefore, the calculation of $\mathbb{D}_i$ can be decomposed into two parts, $\left(M^{\text{Toeplitz}} * M_i\right) \mathbb{Z}_{i+1} h$ and $\left(M^{\text{adjust}} * M_i\right) \mathbb{Z}_{i+1} h$. For the first part, since $M^{\text{Toeplitz}} * M_i$ is a Toeplitz matrix, $\left(M^{\text{Toeplitz}} * M_i\right) \mathbb{Z}_i h$ can be calculated with time complexity $O(m \ln(m))$ via the FFT. For the second part, there are at most 3 non-zero entries in each row of the sparse matrix M^{adjust} (hence $M^{\text{adjust}} * M_i$, too), so $\left(M^{\text{adjust}} * M_i\right) \mathbb{Z}_i h$ can be evaluated in $O(m)$ time. To sum up, we construct a discrete convolution algorithm that converges to the integral convolution at a rate of $O(h^4)$ with time complexity $O(m \ln(m))$.

4.2 Improving the Convergence Rate of the Interpolation Error

The order of the error convergence rate for the interpolation used in line 3 of Algorithm 2 can be improved by the Newton divided-difference interpolation formula (Isaacson and Keller, 1994). From the definition of Z_i in Eq. (10), we have

$$f_{Z_i}(x) = \begin{cases} f_{D_i}(\ln(e^{x+\mu_{i-1}} - 1) - \mu_i) \frac{e^{x+\mu_{i-1}}}{e^{x+\mu_{i-1}} - 1} & \text{if } x > -\mu_{i-1} \\ 0 & \text{otherwise} \end{cases}. \qquad (27)$$

To calculate $\mathbb{Z}_{i,k}$, the approximation of $f_{Z_i}(x_k)$, we first find the corresponding density value $f_{D_i}(\ln(e^{x_k+\mu_{i-1}} - 1) - \mu_i)$. This density value is obtained by interpolating $\mathbb{D}_i$. To make the interpolation error converges at a rate of $O(h^4)$, we first select four grid points $\{x_j, \ldots, x_{j+3}\}$ near $\ln(e^{x+\mu_{i-1}} - 1) - \mu_i$, and then derive a cubic interpolation polynomial $\rho(x)$ that passes through $\{(x_j, \mathbb{D}_{i,j}), \ldots, (x_{j+3}, \mathbb{D}_{i,j+3})\}$. The density value $\mathbb{Z}_{i,k}$ is then calculated by

$$\mathbb{Z}_{i,k} = \begin{cases} \rho(\ln(e^{x_k+\mu_{i-1}} - 1) - \mu_i) \frac{e^{x_k+\mu_{i-1}}}{e^{x_k+\mu_{i-1}} - 1} & \text{if } x_k > -\mu_{i-1} \\ 0 & \text{otherwise} \end{cases}. \qquad (28)$$

4.3 Improving the Error Convergence Rate of the Pricing Results

The 4th-order approximation to the density values f_{D_0} at grid points $x_{-m}, x_{-m+1}, \ldots, x_m$ obtained by the above approach can be used to evaluate the option value by numerically approximating the integral (19). Note that the lower limit $\ln\left(\frac{K(n+1)}{S_{t_0}} - 1\right) - \mu_0$ in the integral may no be in the grid list, making Eq. (19) not directly approximated by Simpson's rule.

To address this problem, we divide Eq. (19) into the principal term

$$\int_{x_{k^*}}^{\infty} f_{D_0}(x) \left(\frac{S_{t_0}}{n+1} \left(1 + e^{x+\mu_0}\right) - K \right) dx \tag{29}$$

and the remainder term

$$\int_{\ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0}^{x_{k^*}} f_{D_0}(x) \left(\frac{S_{t_0}}{n+1} \left(1 + e^{x+\mu_0}\right) - K \right) dx, \tag{30}$$

where k^* stands for the smallest integer such that $x_{k^*} \geq \ln\left(\frac{K(n+1)}{S_{t_0}} - 1\right) - \mu_0$. Note that Eq. (19) equals the sum of Eqs. (29) and (30). The lower limit in Eq. (29) is X_{k^*} instead of $X_{k_0} - h/2$ (see Eq. (20)). This is because we will use Simpson's rule to reduce the integration error, and the settings of lower and upper limits of Simpson's rule and the midpoint rule are different. By ignoring the truncation error, the principal term can be approximated as follows:

$$\int_{x_{k^*}}^{\infty} f_{D_0}(x) \left(\frac{S_{t_0}}{n+1} \left(1 + e^{x+\mu_0}\right) - K \right) dx \approx \int_{x_{k^*}}^{x_m} f_{D_0}(x) \left(\frac{S_{t_0}}{n+1} \left(1 + e^{x+\mu_0}\right) - K \right) dx, \tag{31}$$

where the right-hand side of the equation can be approximated by Simpson's rule and our modified integration rule of Eq. (25). With the 4th-order approximation for f_{D_0} at grid points, applying Simpson's rule, we obtain an approximation for the principal term with error $O(h^4)$. As for the remainder term, we use the definite integral of an interpolating function as an approximation. Specifically, we use the Lagrange interpolating polynomial mentioned in Appendix B to derive a quadratic polynomial $\varphi(x)$ that passes through

$$\left(\ln\left(\frac{K(n+1)}{S_{t_0}} - 1\right) - \mu_0, 0 \right),$$
$$\left(x_{k^*}, \mathbb{D}_{0,k^*} \left(\frac{S_{t_0}}{n+1} \left(1 + e^{x_{k^*}+\mu_0}\right) - K \right) \right),$$
$$\left(x_{k^*+1}, \mathbb{D}_{0,k^*+1} \left(\frac{S_{t_0}}{n+1} \left(1 + e^{x_{k^*+1}+\mu_0}\right) - K \right) \right),$$

and then use

$$\int_{\ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0}^{x_{k^*}} \varphi(x) dx \tag{32}$$

to approximate the remainder term. This approximation has an error bound $O(h^4)$ as shown in Appendix B. Combining 4th-order approximations for the principal and remainder terms, we obtain a 4th-order approximation for the option value.

4.4 Time Complexity of Our Algorithm

$\{\mathbb{R}_i\}$ $(1 \leq i \leq n)$ can be evaluated by the higher-order inverse Fourier transform in $O(nm \ln m)$ time. Each element in list $\mathbb{D}_{n-1}$ is evaluated by applying higher-order interpolation on $\mathbb{R}_{n-1}$ in constant time (Isaacson and Keller, 1994). Thus, it takes $O(m)$ time to evaluate $\mathbb{D}_{n-1}$. Next, $\mathbb{D}_0$ is evaluated by repeating interpolations and convolutions $n-1$ times. Evaluating

$2m+1$ elements in $\mathbb{Z}_i$ by the higher-order interpolation takes $O(m)$ time and convoluting $\mathbb{R}_i$ and $\mathbb{Z}_i$ with our convolution mentioned in last subsection takes $O(m \ln m)$ time. Thus the time complexity to evaluate $\mathbb{D}_0$ is $O(n(m+m\ln m)) = O(nm \ln m)$. Finally, the option value defined in Eq. (1) can be evaluated by the Newton-Cotes integration formulas with $\mathbb{D}_0$ in $O(m)$ time. Thus the total time complexity is $O(nm \ln m)$ time.

4.5 Higher-Order Converging Algorithms

In previous discussions, Simpson's rule (13) and the modified integration rules (25) are used together with the cubic polynomial interpolation formula to improve the error convergence rate from $O(h^2)$ to $O(h^4)$. The error convergence rate can be further improved by applying higher order Newton-Cotes formulas and Newton divided-difference interpolation formulas. For example, Boole's rule and the quintic polynomial interpolation formula can be applied to improve the error convergence rate to $O(h^6)$. Recall that Boole's rule uses quartic polynomial integral to approximate the desired integration with an error convergence rate $O(h^6)$ (see Eq. (14)). Note that the number of sample points is restricted to be $4\kappa+1$ under Boole's rule, where κ denotes an integer. To relax this constraint, three modified integration formulas of Boole's rule are derived by the methodology introduced in Appendix B as follows:

$$\begin{aligned}
\int_{x_j}^{x_j+3h} g(x)dx &= \frac{27g(x_j)+102g(x_j+h)+72g(x_j+2h)+42g(x_j+3h)-3g(x_j+4h)}{80}h \\
\int_{x_j}^{x_j+2h} g(x)dx &= \frac{29g(x_j)+128g(x_j+h)+24g(x_j+2h)+4g(x_j+3h)-g(x_j+4h)}{90}h \\
\int_{x_j}^{x_j+h} g(x)dx &= \frac{251g(x_j)+646g(x_j+h)-264g(x_j+2h)+106g(x_j+3h)-19g(x_j+4h)}{720}h.
\end{aligned} \tag{33}$$

Similarly, the dual version of the the aforementioned integration formulas are listed

$$\begin{aligned}
\int_{x_j+h}^{x_j+4h} g(x)dx &= \frac{-3g(x_j)+42g(x_j+h)+72g(x_j+2h)+102g(x_j+3h)+27g(x_j+4h)}{80}h \\
\int_{x_j+2h}^{x_j+4h} g(x)dx &= \frac{-g(x_j)+4g(x_j+h)+24g(x_j+2h)+128g(x_j+3h)+29g(x_j+4h)}{90}h \\
\int_{x_j+3h}^{x_j+4h} g(x)dx &= \frac{-19g(x_j)+106g(x_j+h)-264g(x_j+2h)+646g(x_j+3h)+251g(x_j+4h)}{720}h.
\end{aligned}$$

Thus the approximated value of $f_{D_{i-1}}(x_k)$, i.e., $\mathbb{D}_{i-1,k}$, can be calculated with Boole's rule and the aforementioned modified integration rule by mimicking the idea to combine the integration rules proposed in Sec. 4.1. For example, $f_{D_{i-1}}(x_{-3})$ can be approximated as follows:

$$\begin{aligned}
& f_{D_{i-1}}(x_{-3}) = \int_{-\infty}^{\infty} f_{R_i}(x_{-3}-x) f_{Z_i}(x)dx \approx \int_{x_{-m}}^{x_{m-3}} f_{R_i}(x_{-3}-x) f_{Z_i}(x)dx \\
= & \int_{x_{-m}}^{x_{-m+1}} f_{R_i}(x_{-3}-x) f_{Z_i}(x)dx + \sum_{j=-m+1,-m+5,\ldots,m-7} \int_{x_j}^{x_{j+4}} f_{R_i}(x_{-3}-x) f_{Z_i}(x)dx \\
\approx & \frac{h}{720}(251\mathbb{R}_{i,m-3}\mathbb{Z}_{i,-m} + 646\mathbb{R}_{i,m-4}\mathbb{Z}_{i,-m+1} - 264\mathbb{R}_{i,m-5}\mathbb{Z}_{i,-m+2} + 106\mathbb{R}_{i,m-6}\mathbb{Z}_{i,-m+3} - 19\mathbb{R}_{i,m-7}\mathbb{Z}_{i,-m+4}) \qquad (34) \\
+ & \sum_{j=-m+1,-m+5,\ldots,m-7} \frac{h}{45}(14\mathbb{R}_{i,j-3}\mathbb{Z}_{i,j} + 64\mathbb{R}_{i,j-4}\mathbb{Z}_{i,j+1} + 24\mathbb{R}_{i,j-5}\mathbb{Z}_{i,j+2} + 64\mathbb{R}_{i,j-6}\mathbb{Z}_{i,j+3} + 14\mathbb{R}_{i,j-7}\mathbb{Z}_{i,j+4}) \qquad (35) \\
\equiv & \mathbb{D}_{i-1,-3},
\end{aligned}$$

where the modified integration rule Eq. (33) is used in Eq. (34), and Boole's rule is used in Eq. (35). Thus, the vector $\mathbb{D}_{i-1}$ can be more accurately evaluated by replacing Simpson's rule used in Eq. (26) with Boole's rule and its modified versions to obtain the convolution $\mathbb{D}_{i-1} = (M^{\mathrm{Boole}} * M)\mathbb{Z}_i h$, where M^{Boole} is defined by

$$\begin{pmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\widehat{\frac{251}{720}} & \overbrace{\frac{646}{720}+\frac{14}{45}} & \overbrace{\frac{-264}{720}+\frac{64}{45}} & \overbrace{\frac{106}{720}+\frac{24}{45}} & \overbrace{\frac{-19}{720}+\frac{64}{45}} & \widehat{\frac{28}{45}} & \cdots & \widehat{\frac{24}{45}} & \widehat{\frac{64}{45}} & \widehat{\frac{14}{45}} & \widehat{0} & \widehat{0} & \widehat{0} \\
\frac{29}{90} & \frac{128}{90} & \frac{24}{90}+\frac{14}{45} & \frac{4}{90}+\frac{64}{45} & \frac{-1}{90}+\frac{24}{45} & \frac{64}{45} & \cdots & \frac{64}{45} & \frac{24}{45} & \frac{64}{45} & \frac{14}{45} & 0 & 0 \\
\frac{27}{80} & \frac{102}{80} & \frac{72}{80} & \frac{42}{80}+\frac{14}{45} & \frac{-3}{80}+\frac{64}{45} & \frac{28}{45} & \cdots & \frac{28}{45} & \frac{64}{45} & \frac{24}{45} & \frac{64}{45} & \frac{14}{45} & 0 \\
\frac{14}{45} & \frac{64}{45} & \frac{24}{45} & \frac{64}{45} & \frac{28}{45} & \frac{64}{45} & \cdots & \frac{64}{45} & \frac{28}{45} & \frac{64}{45} & \frac{24}{45} & \frac{64}{45} & \frac{14}{45} \\
0 & \frac{14}{45} & \frac{64}{45} & \frac{24}{45} & \frac{64}{45} & \frac{28}{45} & \cdots & \frac{28}{45} & \frac{64}{45}+\frac{-3}{80} & \frac{14}{45}+\frac{42}{80} & \frac{72}{80} & \frac{102}{80} & \frac{27}{80} \\
0 & 0 & \frac{14}{45} & \frac{64}{45} & \frac{24}{45} & \frac{64}{45} & \cdots & \frac{64}{45} & \frac{24}{45}+\frac{-1}{90} & \frac{64}{45}+\frac{4}{90} & \frac{14}{45}+\frac{24}{90} & \frac{128}{90} & \frac{29}{90} \\
0 & 0 & 0 & \frac{14}{45} & \frac{64}{45} & \frac{24}{45} & \cdots & \frac{24}{45} & \frac{64}{45}+\frac{-19}{720} & \frac{24}{45}+\frac{106}{720} & \frac{64}{45}+\frac{-264}{720} & \frac{14}{45}+\frac{646}{720} & \frac{251}{720} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{pmatrix}.$$

Note that the values marked with a hat denote the coefficients used for evaluating $\mathbb{D}_{i-1,-3}$ that can be observed from the coefficients of Eqs. (34) and (35). To speed up the evaluation of the aforementioned convolution by taking advantage of the FFT, we take the decomposition $M^{\mathrm{Boole}} = M^{\mathrm{Toeplitz}} + M^{\mathrm{adjust}}$, where

$$M^{\mathrm{Toeplitz}} \equiv \frac{1}{45}\begin{pmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
64 & 28 & 64 & 24 & 64 & 28 & \cdots & 24 & 64 & 14 & 0 & 0 & 0 \\
24 & 64 & 28 & 64 & 24 & 64 & \cdots & 64 & 24 & 64 & 14 & 0 & 0 \\
64 & 24 & 64 & 28 & 64 & 24 & \cdots & 28 & 64 & 24 & 64 & 14 & 0 \\
14 & 64 & 24 & 64 & 28 & 64 & \cdots & 64 & 28 & 64 & 24 & 64 & 14 \\
0 & 14 & 64 & 24 & 64 & 28 & \cdots & 24 & 64 & 28 & 64 & 24 & 64 \\
0 & 0 & 14 & 64 & 24 & 64 & \cdots & 64 & 24 & 64 & 28 & 64 & 24 \\
0 & 0 & 0 & 14 & 64 & 24 & \cdots & 28 & 64 & 24 & 64 & 28 & 64 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{pmatrix}$$

is a Toeplitz matrix, and

$$M^{\mathrm{adjust}} \equiv \begin{pmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
\frac{251}{720} & \frac{646}{720} & -\frac{264}{720} & \frac{106}{720} & -\frac{19}{720} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{29}{90} & \frac{128}{90} & \frac{24}{90} & \frac{4}{90} & -\frac{1}{90} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{27}{80} & \frac{102}{80} & \frac{72}{80} & \frac{42}{80} & -\frac{3}{80} & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\frac{3}{80} & \frac{42}{80} & \frac{72}{80} & \frac{102}{80} & \frac{27}{80} \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\frac{1}{90} & \frac{4}{90} & \frac{24}{90} & \frac{128}{90} & \frac{29}{90} \\
0 & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & -\frac{19}{720} & \frac{106}{720} & -\frac{264}{720} & \frac{646}{720} & \frac{251}{720} \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots
\end{pmatrix}$$

has non-zero values only at the first and last five columns. Recall that the vector $\mathbb{D}_{i-1}$ can be represented as the sum of two matrix multiplications as in Eq. (26). Since $M^{\mathrm{Toeplitz}} * M_i$ is also a Toeplitz matrix, $(M^{\mathrm{Toeplitz}} * M_i)\mathbb{Z}_{i+1}h$ can be calculated in $O(m\ln(m))$ time via the FFT. On the other hand, $(M^{\mathrm{adjust}} * M_i)\mathbb{Z}_{i+1}h$ can be calculated in $O(m)$ time since there are only $O(m)$ non-zero entries in M^{adjust}. To sum up, the vector $\mathbb{D}_{i-1}$ can be evaluated in $O(m\ln(m))$ time. Note that the convergence rate of the integration error is improved to $O(h^6)$ since the error convergence rates of Boole's rule and its modified integration rule are both $O(h^6)$. Similarly, the convergence rate of the interpolation error can be improved to $O(h^6)$ simply by substituting a quintic interpolation polynomial $\rho(x)$ (instead of a cubic one) into Eq. (28). By repeatedly applying the aforementioned convolution method and the interpolation method, we can obtain the

6th-order approximation to f_{D_0} in $O(nm\ln(m))$ time. To ensure that the pricing error also converges in $O(h^6)$, we first recall that the option value in Eq. (19) can be decomposed into the principal term in Eq. (29) and the remainder term in Eq. (30). The principal term Eq. (29) can be rewritten as Eq. (31) and then approximated by Boole's rule and our modified 6th-order integration rule. To estimate the remainder term Eq. (30), we first derive a quartic interpolating polynomial $\varphi(x)$ that passes through $\left(\ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0, 0\right)$, $\left(x_{k^*}, \mathbb{D}_{0,k^*}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_{k^*}+\mu_0}\right)-K\right)\right)$, $\left(x_{k^*+1}, \mathbb{D}_{0,k^*+1}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_{k^*+1}+\mu_0}\right)-K\right)\right)$, $\left(x_{k^*+2}, \mathbb{D}_{0,k^*+2}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_{k^*+2}+\mu_0}\right)-K\right)\right)$, and $\left(x_{k^*+3}, \mathbb{D}_{0,k^*+3}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_{k^*+3}+\mu_0}\right)-K\right)\right)$ by the Lagrange interpolating polynomial mentioned in Appendix B. Then the remainder term is approximated by integrating $\varphi(x)$ as in Eq. (32). Since the error convergence rates of these two approximations are $O(h^6)$, the pricing error of our method converges at a rate of $O(h^6)$.

5 Numerical Results

To compare the error convergence rates among the pricing algorithms discussed in this paper, the log-log plots on the absolute pricing errors against h are employed. Specifically, an $O(h^k)$ error convergence rate implies that the pricing error $e(h)$ can be represented as

$$e(h) \leq \tilde{c}h^k,$$

where $\tilde{c}$ denotes a constant. To numerically estimate the error convergence rate of an arbitrary algorithm, we first plot the points $(\ln(h), \ln(e(h)))$ of that algorithm with various h. The estimated error convergence rate is the slope of the linear regression line for these points.

If the stock price follows LGNO and the strike price $K = 0$, the Asian call option can be priced analytically (Dai et al., 2005). Thus we can calculate exact pricing errors and compare the error convergence rate between the Benhamou algorithm and our $O(h^4)$ and $O(h^6)$ algorithms in panel (a) of Figure 2. The slopes of the linear regression functions are 2.07429, 4.2457, and 6.13852, respectively, which implies that the error convergent rates for these three algorithms are roughly $O(h^2)$, $O(h^4)$ and $O(h^6)$, respectively.

Analytical formulas are unavailable under the non-zero strike price case. Figure 3 shows that the pricing results of the three algorithms converge to the same value as the number of grid points m tends to infinity. Panel (a) plots the pricing results of the three algorithms. Note that the Benhamou algorithm (denoted by circles) converges much more slowly than the other two algorithms. Panel (b) suggests our 6th-order algorithm (denoted by squares) converges faster than our 4th-order algorithm (denoted by triangles). The plot for $\ln(h)$ against the natural logarithm of the absolute pricing error is illustrated in panel (b) of Figure 2, where the exact option value is obtained by using the extrapolation method proposed in Heston and Zhou (2000) on the numerical results in Figure 3. The slopes of the linear regression functions for the three algorithms are 2.07217, 4.24851, 6.03232, respectively, which again implies that the error convergence rates for the three algorithms are $O(h^2)$, $O(h^4)$ and $O(h^6)$, respectively.

When n is large enough, our algorithms also converge to the price of a continuously monitored Asian option. The price of a continuously monitored Asian option has been widely studied, and accurate pricing results under the LGNO model can be evaluated (see, for example, Milevsky and Posner (1998); Zhang (2003)). We use the pricing results suggested by Zhang (2003) as the benchmarks and compare them with our 4th-order algorithm in Table 2. As shown in the table, for all parameter settings, the pricing results generated by our algorithm approximate the benchmarks very well.

Furthermore, our algorithm also converges smoothly and fast under the NIG, the DE, and the JD models as illustrated in panel (a), (b), and (c) of Figure 4, respectively. While the pricing results generated by the Benhamou algorithm (plotted in circles) oscillate significantly, the results of ours (plotted in squares) converge steadily.

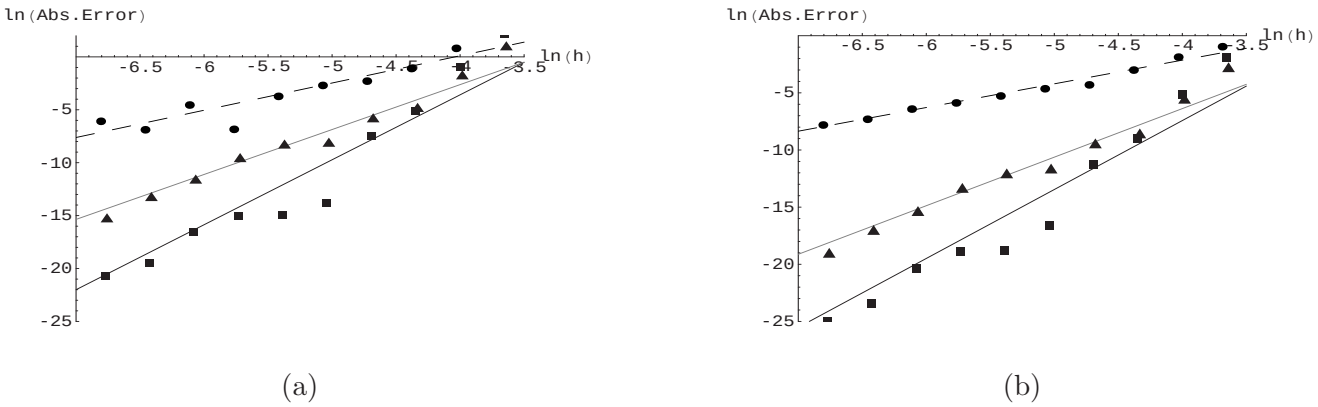
The running time comparison for the aforementioned three algorithms are illustrated in Figure 5, where the x-axis and y-axis denote $\ln(m\ln(m))$ and the natural logarithm of the running time, respectively. Recall that the running time complexities for the three algorithms are all $O(nm\ln(m))$. In other words, the running time $t(m)$ can be represented as

$$t(m) \approx \hat{c}nm\ln(m),$$

where $\hat{c}$ denotes a constant.[2] By taking natural logarithm on both side, we have

$$t(m) \approx \ln(\hat{c}n) + \ln(m\ln(m)).$$

That is, the plot of $\ln(m\ln(m))$ against $\ln(t(m))$ should approximate a straight line with slope=1, and this is confirmed in Figure 5. In addition, a higher-order algorithm is more accurate than a lower-order one with the same number of grid points m as illustrated in Figure 3 but they use almost the same running time as illustrated in Figure 5. Thus we conclude that our approach does significantly improve the efficiency.



(a) (b)

Figure 2: Convergence of Pricing Results: Log-Log Plots.
The log-log plots for the pricing results are shown in panel (a) with parameters $r = 0.1, S_{t_0} = 100, \sigma = 0.3, T = 1, n = 50, K = 0$. Panel (b) gives the log-log plot under the same settings except that $K = 120$. The x-axis and the y-axis denote $\ln h$ and the natural logarithm of the absolute pricing error, respectively. In panel (a) we use the closed form pricing formula given by Dai et al. (2005) to evaluate the exact value needed for calculating the pricing errors. The value 2.36828545183 obtained by the extrapolation method proposed by Heston and Zhou (2000) is used as a proxy for the true option value in panel (b). In both panels, the circles, the triangles and the squares denote the results computed by the Benhamou algorithm, our 4th-order and 6th-order pricing algorithms, respectively. The dashed line, the gray line, and the solid black line denote the regression lines for the three algorithms.

[2]Note that the number of stock price samples n is predetermined in the option contract. We assume that $n = 50$ in this case.

		$\sigma = 0.2$		$\sigma = 0.3$	
K	r	Zhang	ours	Zhang	ours
90		12.596	12.595	13.952	13.952
100	0.05	5.763	5.762	7.946	7.944
110		1.990	1.989	4.072	4.070
90		13.831	13.831	14.984	14.983
100	0.09	6.777	6.776	8.829	8.827
110		2.546	2.545	4.697	4.695
90		15.642	15.642	16.513	16.512
100	0.15	8.409	8.408	10.210	10.208
110		3.556	3.555	5.730	5.729
		RMSE	0.001	RMSE	0.002

Table 2: Compare to continuously sampled Asian option price.
This table shows that our pricing results converge to the prices of the continuously sampled Asian options as n large enough, where "Zhang" denotes the pricing results given by Zhang (2003) with $S_0 = 100$, $T = 1$, and our pricing results are evaluated by the 4th-order converging algorithm with re-centering, where the numerical settings are $c = 1, n = 800$, and $h \approx 1.56 \times 10^{-4}$.

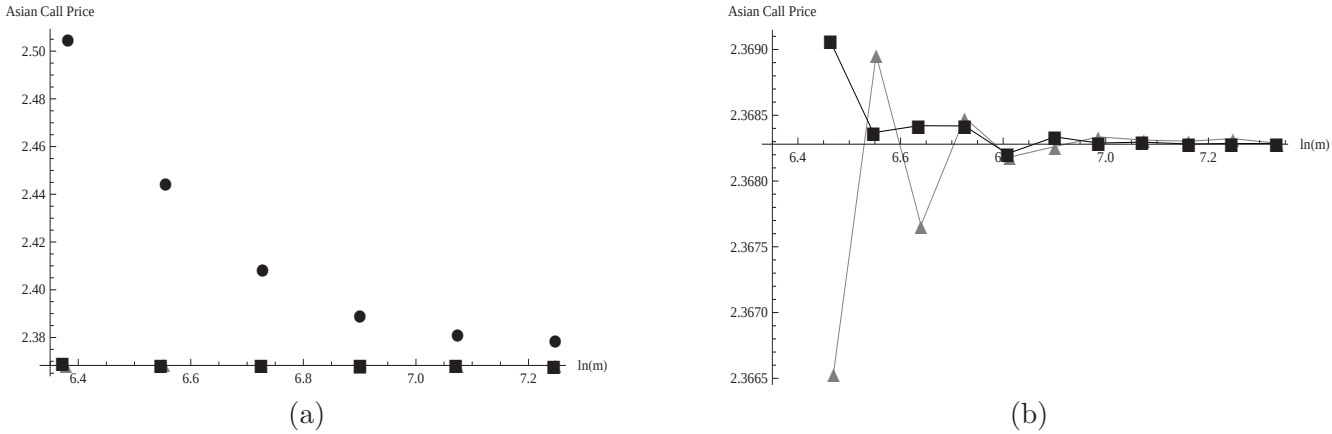
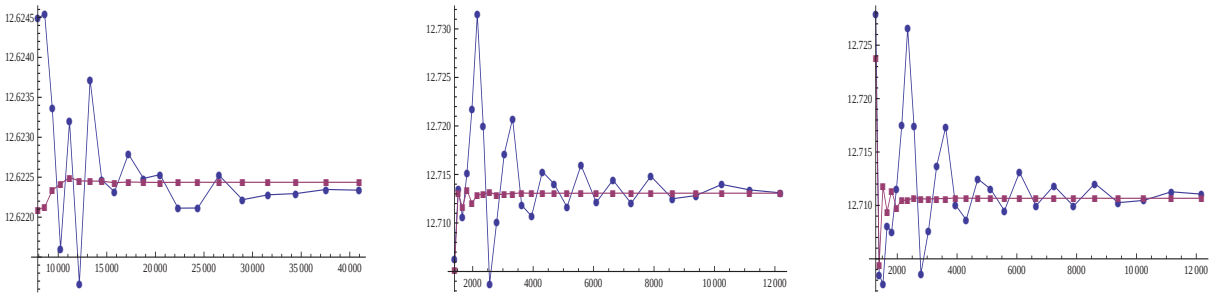


Figure 3: Convergence of Pricing Results for the Non-Zero Strike Price Case.
The settings are the same as in Figure 2 (b). The x- and the y-axes denote $\ln m$ and the pricing results, respectively. The circles, the triangles, and the squares denote the results computed by the Benhamou algorithm, our 4th-order and 6th-order algorithms, respectively.

6 Conclusion

Asian options can be priced numerically by estimating the density of the payoff function with repeated interpolation and convolution. Previous algorithms use the midpoint rule to approximate the convolution, which can be sped up by the FFT. We show that the pricing error of their algorithms converges quadratically. To improve the error convergence rate, we reduce the integration error by higher-order Newton-Cotes integration formulas, like Simpson's rule and Boole's rule, and the interpolation error by higher-order Newton divided-difference interpolation. We also derive the modified integration rule to bypass the constraint on the number of



(a) (b) (c)

Figure 4: Convergence of Pricing Results under Lévy Processes.
Numerical settings are $r = 0.0367$, $S_{t_0} = 100$, $K = 90$, $T = 1$, $n = 12$, and $c = 8$. Panel (a), (b) and (c) show the pricing results under the NIG, the DE, and the JD model, respectively. The numerical settings under the NIG model are $\alpha = 6.1882$, $\beta = -3.8941$, $\delta = 0.1622$; under the DE model, the settings are $\sigma = 0.120381$, $p = 0.2071$, $\lambda = 0.330966$, $\eta_2 = 3.13868$, $\eta_1 = 9.65997$; under the JD model, the settings are $\sigma = 0.126349$, $\alpha = -0.390078$, $\lambda = 0.174814$, $\delta = 0.338796$. The x-axis denotes m and the y-axis denotes the pricing results. The circles and the squares denote the results computed by the Benhamou algorithm and our 4th-order algorithm, respectively.

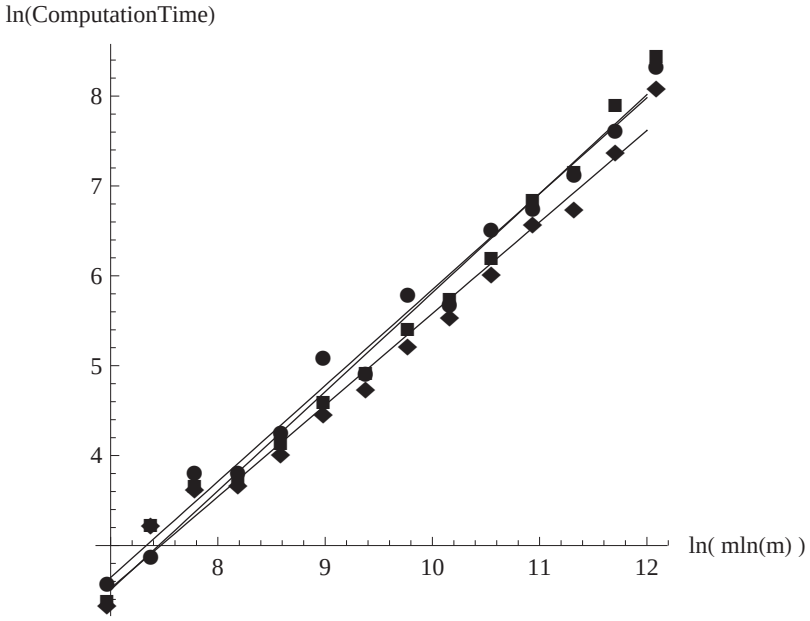


Figure 5: Running Time Comparison.
The numerical settings are the same as in Figure 2 (b). The x-axis and the y-axis denote the natural logarithm of $\ln(m \ln(m))$ and the natural logarithm of the running time, respectively. The circles, the triangles and the squares denote the results computed by the Benhamou algorithm, our 4th-order and 6th-order pricing algorithms, respectively. The dashed line, the gray line, and the solid black line denote the regression lines for the three algorithms.

sample points required by the higher-order Newton-Cotes formulas. The resulting convolution can be sped up by the FFT. Thus our algorithm can achieve the same time complexity as previously announced algorithms, but at a faster error convergence rate. Numerical experiments show the superiority of our algorithm to price Asian options under some Lévy processes that

are frequently used in the literature.

References

Aingworth, D., R. Motwani, and J. Oldham (2000). Accurate approximations for Asian options. In *Proceedings of the 11th ACM-SIAM SODA*, pp. 891–900.

Benhamou, E. (2002). Fast Fourier transform for discrete Asian options. *Journal of Computational Finance 6*(1), 49–68.

Boyle, P., M. Broadie, and P. Glasserman (1997). Monte Carlo methods for security pricing. *Journal of Economic Dynamics and Control 21*(8–9), 1267–1321.

Broadie, M., P. Glasserman, and S. Kou (1999). Connecting discrete and continuous path-dependent options. *Finance and Stochastics 3*(1), 55–82.

Carverhill, A. and L. Clewlow (1990). Flexible convolution. *Risk 3*(4), 25–29.

Černỳ, A. and I. Kyriakou (2011). An improved convolution algorithm for discretely sampled Asian options. *Quantitative Finance 11*(3), 381–389.

Cormen, T., C. Leiserson, R. Rivest, and C. Stein (2001). *Introduction to algorithms, 2nd ed.* Cambridge, MA: MIT Press.

Dai, T., G. Huang, and Y. Lyuu (2005). An efficient convergent lattice algorithm for European Asian options. *Applied Mathematics and Computation 169*(2), 1458–1471.

Fusai, G. and A. Meucci (2008). Pricing discretely monitored Asian options under Lévy processes. *Journal of Banking & Finance 32*(10), 2076–2088.

Harrison, J. and S. Pliska (1981). Martingales and stochastic integrals in the theory of continuous trading. *Stochastic Processes and Their Applications 11*(3), 215–260.

Heston, S. and G. Zhou (2000). On the rate of convergence of discrete-time contingent claims. *Mathematical Finance 10*(1), 53–75.

Hosking, J., G. Bonti, and D. Siegel (2000). Beyond the lognormal. *Risk 13*(5), 59–62.

Huisman, R., K. Koedijk, and R. Pownall (1998). Var-x: Fat tails in financial risk management. *Journal of Risk 1*(1), 47–61.

Ingersoll, J. (1987). *Theory of financial decision making.* Savage, Maryland, Rowman & Littlefield.

Isaacson, E. and H. Keller (1994). *Analysis of numerical methods.* New York, Dover.

Kemna, A. and A. Vorst (1990). A pricing method for options based on average asset values. *Journal of Banking & Finance 14*(1), 113–129.

Levy, E. (1992). Pricing European average rate currency options. *Journal of International Money and Finance 11*(5), 474–491.

Milevsky, M. and S. Posner (1998). Asian options, the sum of lognormals, and the reciprocal gamma distribution. *Journal of Financial and Quantitative Analysis 33*(2), 409–422.

Turnbull, S. and L. Wakeman (1991). A quick algorithm for pricing European average options. *Journal of Financial and Quantitative Analysis 26*(3), 377–389.

Zhang, J. (2003). Pricing continuously sampled Asian options with perturbation method. *Journal of Futures Markets 23*(6), 535–560.

Zhang, P. (1998). *Exotic options*. Singapore, World Scientific.

A The Derivation of $\mathbb{R}_i$

Given $\varphi_{R_i}(u) \equiv E(e^{\mathbf{i}uR_i})$ in analytical form, our goal is to find the approximating list $\mathbb{R}_{i,k} \approx f_{R_i}(x_k)$ for all $k = -m, -m+1, \ldots, m$. Since $\varphi_{R_i}(u)$ is the Fourier transform of $f_{R_i}(x)$:

$$\varphi_{R_i}(u) = \int_{-\infty}^{\infty} e^{\mathbf{i}ux} f_{R_i}(x) dx,$$

to get $f_{R_i}(x)$ we apply the inverse Fourier transform:

$$f_{R_i}(x) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-\mathbf{i}xu} \varphi_{R_i}(u) du.$$

For some special $\varphi_{R_i}(u)$, this integral does have an explicit formula, but if such a formula does not exist, numerical schemes are necessary. The above integral can be approximated by the Newton-Cotes integration as follows:

$$f_{R_i}(x_k) \approx \frac{\dot{h}}{2\pi} \sum_j e^{-\mathbf{i}x_k u_j} \bar{w}_j \varphi_{R_i}(u_j), \tag{36}$$

where $\{u_j\}$ is an equal-distanced grid list with $u_j - u_{j-1} = \dot{h}$, and $\{\bar{w}_j\}$ denotes the Newton-Cotes coefficients, which govern the error convergence rate of this approximation. For convenience, we set $\bar{w}_j = 1$ for all j below, which implies the midpoint rule is used. Since Eq. (36) is similar to the inverse discrete Fourier transform, we try to speed up its computation by the FFT.

The FFT computes the matrix multiplication $F\vec{v}$ in $O(m \ln m)$ time (Cormen et al., 2001), where $\vec{v}$ is an arbitrary $(2m+1) \times 1$ vector, and

$$F \equiv \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ 1 & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 1\cdot 1} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 1\cdot 2} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 1\cdot 3} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 1\cdot 4} & \cdots & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 1(2m)} \\ 1 & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 2\cdot 1} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 2\cdot 2} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 2\cdot 3} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 2\cdot 4} & \cdots & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 2(2m)} \\ 1 & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 3\cdot 1} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 3\cdot 2} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 3\cdot 3} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 3\cdot 4} & \cdots & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 3(2m)} \\ 1 & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 4\cdot 1} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 4\cdot 2} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 4\cdot 3} & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 4\cdot 4} & \cdots & e^{-\frac{2\pi\mathbf{i}}{2m+1}\cdot 4(2m)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & e^{-\frac{2\pi\mathbf{i}}{2m+1}(2m)1} & e^{-\frac{2\pi\mathbf{i}}{2m+1}(2m)2} & e^{-\frac{2\pi\mathbf{i}}{2m+1}(2m)3} & e^{-\frac{2\pi\mathbf{i}}{2m+1}(2m)4} & \cdots & e^{-\frac{2\pi\mathbf{i}}{2m+1}(2m)(2m)} \end{pmatrix}.$$

To match this special matrix, the exponent of Eq. (36), $-\mathbf{i}x_k u_j$, should equal $-\frac{2\pi\mathbf{i}}{2m+1}kj$. Since $x_k = kh$ is predetermined, it follows that

$$u_j \equiv \frac{2\pi j}{(2m+1)h},$$

and hence $\dot{h} \equiv 2\pi/((2m+1)h)$. For notational simplicity, let $\nu \equiv -\frac{2\pi\mathbf{i}}{2m+1}$, and $\chi \equiv \frac{2\pi}{2m+1}$. Substituting $x_k = kh, u_j = \frac{2\pi j}{(2m+1)h}$ into Eq. (36) and rewriting it as a matrix multiplication,

we obtain

$$\begin{pmatrix} f_{R_i}(-mh) \\ \vdots \\ f_{R_i}(-2h) \\ f_{R_i}(-h) \\ f_{R_i}(0) \\ f_{R_i}(h) \\ f_{R_i}(2h) \\ \vdots \\ f_{R_i}(mh) \end{pmatrix} \approx \frac{\dot{h}}{2\pi} \underbrace{\begin{pmatrix} e^{(-m)(-m)\nu} & \dots & e^{(-m)(-2)\nu} & e^{(-m)(-1)\nu} & e^{(-m)(0)\nu} & e^{(-m)(1)\nu} & e^{(-m)(2)\nu} & \dots & e^{(-m)(m)\nu} \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ e^{(-2)(-m)\nu} & \dots & e^{(-2)(-2)\nu} & e^{(-2)(-1)\nu} & e^{(-2)(0)\nu} & e^{(-2)(1)\nu} & e^{(-2)(2)\nu} & \dots & e^{(-2)(m)\nu} \\ e^{(-1)(-m)\nu} & \dots & e^{(-1)(-2)\nu} & e^{(-1)(-1)\nu} & e^{(-1)(0)\nu} & e^{(-1)(1)\nu} & e^{(-1)(2)\nu} & \dots & e^{(-1)(m)\nu} \\ e^{(0)(-m)\nu} & \dots & e^{(0)(-2)\nu} & e^{(0)(-1)\nu} & e^{(0)(0)\nu} & e^{(0)(1)\nu} & e^{(0)(2)\nu} & \dots & e^{(0)(m)\nu} \\ e^{(1)(-m)\nu} & \dots & e^{(1)(-2)\nu} & e^{(1)(-1)\nu} & e^{(1)(0)\nu} & e^{(1)(1)\nu} & e^{(1)(2)\nu} & \dots & e^{(1)(m)\nu} \\ e^{(2)(-m)\nu} & \dots & e^{(2)(-2)\nu} & e^{(2)(-1)\nu} & e^{(2)(0)\nu} & e^{(2)(1)\nu} & e^{(2)(2)\nu} & \dots & e^{(2)(m)\nu} \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ e^{(m)(-m)\nu} & \dots & e^{(m)(-2)\nu} & e^{(m)(-1)\nu} & e^{(m)(0)\nu} & e^{(m)(1)\nu} & e^{(m)(2)\nu} & \dots & e^{(m)(m)\nu} \end{pmatrix}}_{\tilde{F}} \begin{pmatrix} \varphi_{R_i}\left(\frac{-m}{h}\chi\right) \\ \vdots \\ \varphi_{R_i}\left(\frac{-2}{h}\chi\right) \\ \varphi_{R_i}\left(\frac{-1}{h}\chi\right) \\ \varphi_{R_i}\left(\frac{0}{h}\chi\right) \\ \varphi_{R_i}\left(\frac{1}{h}\chi\right) \\ \varphi_{R_i}\left(\frac{2}{h}\chi\right) \\ \vdots \\ \varphi_{R_i}\left(\frac{m}{h}\chi\right) \end{pmatrix}.$$

The elements of the above matrix $\tilde{F}$ are exactly the same as those of F, but different in order. To see this, recall that due to the nature of the exponential function, we have $e^0 = 1$ and

$$e^{\frac{\mathbf{i}2\pi}{2m+1}(k)(j)} = e^{\frac{\mathbf{i}2\pi}{2m+1}(k+2m+1)(j)} = e^{\frac{\mathbf{i}2\pi}{2m+1}(k)(j+2m+1)} = e^{\frac{\mathbf{i}2\pi}{2m+1}(k+2m+1)(j+2m+1)}.$$

Hence, the $\tilde{F}$ can be rewritten as

$$\begin{pmatrix} e^{(m+1)(m+1)\nu} & \dots & e^{(m+1)(2m-1)\nu} & e^{(m+1)(2m)\nu} & 1 & e^{(m+1)(1)\nu} & e^{(m+1)(2)\nu} & \dots & e^{(m+1)(m)\nu} \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ e^{(2m-1)(m+1)\nu} & \dots & e^{(2m-1)(2m-1)\nu} & e^{(2m-1)(2m)\nu} & 1 & e^{(2m-1)(1)\nu} & e^{(2m-1)(2)\nu} & \dots & e^{(2m-1)(m)\nu} \\ e^{(2m)(m+1)\nu} & \dots & e^{(2m)(2m-1)\nu} & e^{(2m)(2m)\nu} & 1 & e^{(2m)(1)\nu} & e^{(2m)(2)\nu} & \dots & e^{(2m)(m)\nu} \\ 1 & \cdots & 1 & 1 & 1 & 1 & 1 & \cdots & 1 \\ e^{(1)(m+1)\nu} & \dots & e^{(1)(2m-1)\nu} & e^{(1)(2m)\nu} & 1 & e^{(1)(1)\nu} & e^{(1)(2)\nu} & \dots & e^{(1)(m)\nu} \\ e^{(2)(m+1)\nu} & \dots & e^{(2)(2m-1)\nu} & e^{(2)(2m)\nu} & 1 & e^{(2)(1)\nu} & e^{(2)(2)\nu} & \dots & e^{(2)(m)\nu} \\ \vdots & & \vdots & \vdots & \vdots & \vdots & \vdots & & \vdots \\ e^{(m)(m+1)\nu} & \dots & e^{(m)(2m-1)\nu} & e^{(m)(2m)\nu} & 1 & e^{(m)(1)\nu} & e^{(m)(2)\nu} & \dots & e^{(m)(m)\nu} \end{pmatrix}.$$

Note that $\tilde{F} = P_1 F P_2$, where P_1 and P_2 are permutation matrices given by

$$P_1 \equiv \begin{pmatrix} 0 & I_m \\ I_{m+1} & 0 \end{pmatrix}, \quad P_2 \equiv \begin{pmatrix} 0 & I_{m+1} \\ I_m & 0 \end{pmatrix},$$

where I_κ denotes the $\kappa \times \kappa$ identity matrix.

The evaluation of $\mathbb{R}_i$ is now clear:

$$\mathbb{R}_i = \frac{\dot{h}}{2\pi} P_1 F P_2 \vec{\varphi_{R_i}},$$

where $\vec{\varphi_{R_i}}$ represents the column vector $(\varphi_{R_i}(u_{-m}), \dots, \varphi_{R_i}(u_{-1}), \varphi_{R_i}(u_0), \varphi_{R_i}(u_1), \dots, \varphi_{R_i}(u_m))'$. Since P_1, P_2 are both sparse matrices, and the computation of multiplying F by a vector can be sped up by the FFT, the total time complexity to evaluate $\mathbb{R}_i$ is $O(m \ln m)$. Furthermore, as an approximation for $f_{R_i}(x_k)$, the error convergence rate of $\mathbb{R}_i$ can be reduced by replacing $\vec{\varphi_{R_i}}$ with $(w_{-m}^{-}\varphi_{R_i}(u_{-m}), \dots, w_{-1}^{-}\varphi_{R_i}(u_{-1}), \bar{w_0}\varphi_{R_i}(u_0), w_{-1}^{-}\varphi_{R_i}(u_1), \dots, \bar{w_m}\varphi_{R_i}(u_m))'$. Although there may have certain constrain on the number of terms of Eq. (36), we can still relax it by the modified integration formula derived in Appendix B.

B Derive General Numerical Integration Formulas Given the Error Convergence Rates

To integrate a smooth, but unknown function $g(t)$ given the function's values at some certain points, we will first derive an approximating polynomial $\phi(t)$ by the Lagrange interpolating polynomial and then derive the numerical integration formula by integrating $\phi(t)$ instead. The numerical integration can be reinterpreted as the weighted average values of $g(t)$ at the points and the error convergence rate can be represented as a function of number of the points and the distances between the points. A general form for this numerical integration formula will be first introduced. Then we will show how Eq. (25) is derived and how Eq. (32) is integrated numerically by this general form. Finally, we will show that the pricing error for neglecting the remainder term in Benhamou's algorithm (see Sec. 3.1) converges quadratically.

Given the values of $g(t)$ at arbitrary $r-1$ points $t_1 < t_2 < \cdots < t_{r-1}$, the definite integral of $g(t)$ over a given interval $[t_i, t_j] \subseteq [t_1, t_{r-1}]$ can be approximated by the integral of the polynomial $\phi(t)$ that pass through $(t_1, g(t_1)), (t_2, g(t_2)), \ldots, (t_{r-1}, g(t_{r-1}))$. This $\phi(t)$ can be generated by the Lagrange interpolating polynomial as follows:

$$\phi(t) = \sum_{s=1}^{r-1} g(t_s) L_s(t),$$

where $L_s(t)$ is the Lagrange basis polynomial defined as follows:

$$L_s(t) = \prod_{l=1, l\neq s}^{r-1} \frac{t - t_l}{t_s - t_l}. \tag{37}$$

As $t \in [t_1, t_{k-1}]$, the error between $g(t)$ and $\phi(t)$ is bounded above by $O(k^{r-1})$(Isaacson and Keller, 1994), where $k = \max_{2\leq l\leq r-1}(t_l - t_{l-1})$. That is, $|\phi(t) - g(t)| \leq ck^{r-1}$, where c denotes a constant. Thus the integration error is

$$\left| \int_{t_i}^{t_j} \phi(t)dt - \int_{t_i}^{t_j} g(t)dt \right| \leq \int_{t_i}^{t_j} |\phi(t) - g(t)|\, dt \leq \int_{t_i}^{t_j} ck^{r-1} dt = c(t_j - t_i)k^{r-1} \leq c(j-i)k^r = O(k^r).$$

Furthermore, since $\phi(t)$ is a linear combination of the Lagrange basis polynomials, the integral of $\phi(t)$ can be rewritten as the weighted average values of $g(t)$ as follows:

$$\int_{t_i}^{t_j} \phi(t)dt = \int_{t_i}^{t_j} \left(\sum_{s=1}^{r-1} g(t_s) L_s(t) \right) dt = \sum_{s=1}^{r-1} \left(\int_{t_i}^{t_j} L_s(t)dt \right) g(t_s). \tag{38}$$

Two special examples are given below to demonstrate how the aforementioned general forms are used to derive numerical integration formulas. We first consider the case that $t_1, t_2, \ldots, t_{r-1}$ are equidistant grid points. Assume that we want to derive an integration formula with an error convergence rate $O(k^4)$ as in Eq. (25). We first substitute $r = 4$ and $t_3 - t_2 = t_2 - t_1 = k$ into Eq. (37) to obtain

$$\begin{aligned}
L_1(t) &= \frac{(t-t_2)(t-t_3)}{(t_1-t_2)(t_1-t_3)} = \frac{(t-t_1-k)(t-t_1-2k)}{2k^2}, \\
L_2(t) &= \frac{(t-t_3)(t-t_1)}{(t_2-t_3)(t_2-t_1)} = \frac{(t-t_1-2k)(t-t_1)}{-k^2}, \\
L_3(t) &= \frac{(t-t_1)(t-t_2)}{(t_3-t_1)(t_3-t_2)} = \frac{(t-t_1)(t-t_1-k)}{2k^2}.
\end{aligned}$$

Then we substitute the above identities into Eq. (38) to obtain[3]

$$\sum_{s=1}^{3}\left(\int_{t_1}^{t_1+k} L_s(t)dt\right) g(t_s) = \frac{5g(t_1)+8g(t_2)-g(t_3)}{12}k$$

and

$$\sum_{s=1}^{3}\left(\int_{t_3-k}^{t_3} L_s(t)dt\right) g(t_s) = \frac{-g(t_1)+8g(t_2)+5g(t_3)}{12}k,$$

which derives Eq. (25). Other integration formulas introduced in section 4 can also be derived by the same approach.

Consider the case that the distances between adjacent grid points are not equal, like the integration for the remainder term in Eq. (32). Let k and $\overline{k}$ stand for $t_3 - t_2$ and $t_2 - t_1$, respectively. Note that $k > \overline{k}$. To ensure an $O(k^4)$ error convergence rate, we substitute $r = 4$, k, and $\overline{k}$ into Eqs. (37) and (38) to obtain

$$\sum_{s=1}^{3}\left(\int_{t_1}^{t_2} L_s(t)dt\right) g(t_s) = \frac{k\left(\overline{k}(3\overline{k}+2k)g(t_1) + \left(3\overline{k}^2+4k\overline{k}+k^2\right)g(t_2) - k^2 g(t_3)\right)}{6\overline{k}(\overline{k}+k)}.$$

Thus Eq. (32) can be numerically solved by substituting $t_1 = \ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0, t_2 = x_{k^*}, t_3 = x_{k^*+1}$ and $g(t_1) = 0, g(t_2) = \mathbb{D}_{0,k^*}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_{k^*}+\mu_0}\right)-K\right), g(t_3) = \mathbb{D}_{0,k^*+1}\left(\frac{S_{t_0}}{n+1}\left(1+e^{x_{k^*+1}+\mu_0}\right)-K\right)$ into the above formula.

The aforementioned analysis can be used to show that the pricing error caused by neglecting the remainder term (see Eq. (21)) in Benhamou's algorithm converges quadratically. Substituting $r = 3$ and $\overline{k}$ into Eqs. (37) and (38) yields the Trapezoidal Rule:

$$\sum_{s=1}^{2}\left(\int_{t_1}^{t_2} L_s(t)dt\right) g(t_s) = \frac{g(t_1)+g(t_2)}{2}\overline{k},$$

which is an 3rd-order approximation for $\int_{t_1}^{t_2} g(t)dt$. Note that $g(t_1) = 0$ and $g(t_2)$ can be expressed as $g(t_1) + \overline{k}g'(\xi)$ by Taylor expansion, where $\xi \in [t_1, t_2]$. We have

$$\int_{t_1}^{t_2} g(t)dt = \frac{g(t_1)+g(t_2)}{2}\overline{k} + O(\overline{k}^3) = \frac{g'(\xi)}{2}\overline{k}^2 + O(\overline{k}^3).$$

Assume that

$$\begin{aligned} t_1 &= \ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0, \\ t_2 &= x_{k_0} - \frac{h}{2} = t_1 + \bar{k}, \\ g(t) &= f_{D_0}(t)\left(\frac{S_{t_0}}{n+1}\left(1+e^{t+\mu_0}\right)-K\right). \end{aligned}$$

Thus $\int_{t_1}^{t_2} g(t)dt$ is reduced to the remainder term given in Eq. (21), which can now be rewritten as

$$\begin{aligned} & \frac{g(t_1)+g(t_2)}{2}\bar{k} + O(\bar{k}^3) \\ = & \frac{0 + f_{D_0}\left(x_{k_0}-\frac{h}{2}\right)\left(\frac{S_{t_0}}{n+1}\left(1+e^{t_1+\bar{k}+\mu_0}\right)-K\right)}{2}\bar{k} + O(\bar{k}^3) \\ = & \frac{f_{D_0}\left(x_{k_0}-\frac{h}{2}\right)}{2}\left(\frac{S_{t_0}}{n+1}\left(1+e^{t_1+\mu_0}(1+O(\bar{k}))\right)-K\right)\bar{k} + O(\bar{k}^3) \end{aligned}$$

[3]For example, $\int_{t_1}^{t_1+k} L_1(t)dt = \int_0^k \frac{(u-k)(u-2k)}{2k^2}du = \frac{5k}{12}$, where we use the substitution $u = t - t_1$. Other coefficients can also be evaluated by a similar technique.

Since $\frac{S_{t_0}}{n+1}\left(1+e^{t_1+\mu_0}\right)-K=0$, the above expression can be reduced to

$$\begin{aligned}&\frac{f_{D_0}\left(x_{k_0}-\frac{h}{2}\right)}{2}\left(\frac{S_{t_0}O(\bar{k})e^{t_1+\mu_0}}{n+1}\right)\bar{k}+O(\bar{k}^3)\\ =\;&\frac{f_{D_0}\left(x_{k_0}-\frac{h}{2}\right)}{2}\left(\frac{S_{t_0}e^{t_1+\mu_0}}{n+1}\right)O(\bar{k}^2)+O(\bar{k}^3)\end{aligned}$$

.

Substituting $t_1=\ln\left(\frac{K(n+1)}{S_{t_0}}-1\right)-\mu_0$ again into the above expression we obtain

$$\frac{f_{D_0}\left(x_{k_0}-\frac{h}{2}\right)}{2}\left(K-\frac{S_{t_0}}{n+1}\right)O(\bar{k}^2)+O(\bar{k}^3)=O(\bar{k}^2),$$

provided $\frac{f_{D_0}\left(x_{k_0}-\frac{h}{2}\right)}{2}\left(K-\frac{S_{t_0}}{n+1}\right)$ is bounded above.