The Journal of Computational Finance (1–26) Volume 12/Number 1, Fall 2008

Multi-asset option pricing using a parallel Fourier-based technique

C. C. W. Leentvaar
Delft University of Technology, Numerical Analysis Group, Mekelweg 4, 2628CD Delft, The Netherlands; email: c.c.w.leentvaar@tudelft.nl

C. W. Oosterlee
Delft University of Technology, Numerical Analysis Group, Mekelweg 4, 2628CD Delft, The Netherlands
and
CWI, Center for Mathematics and Computer Science, Amsterdam, The Netherlands; email: c.w.oosterlee@cwi.nl

In this paper we present and evaluate a Fourier-based sparse grid method for pricing multi-asset options. This involves computing multi-dimensional integrals efficiently and we accomplish it using the fast Fourier transform. We also propose and evaluate ways to deal with the curse of dimensionality by means of parallel partitioning of the Fourier transform and by incorporating a parallel sparse grid method. Finally, we test the presented method by solving equations for options that are dependent on up to seven underlying assets.

1 INTRODUCTION

A parallel method for pricing multi-asset options is presented in this paper. The core of the method is the computation of a multi-dimensional integral. By means of the fast Fourier transform (FFT) we can compute this integral efficiently. Deterministic solvers (as opposed to Monte Carlo methods) on tensor-product grids for multi-dimensional problems, however, suffer from the so-called "curse of dimensionality", ie, the exponential increase in the number of unknowns as the dimension d increases. This is also true for the FFT-based technique presented here. In order to reduce the complexity we combine a parallel partitioning method for the Fourier transform with a sparse grid method. The method is evaluated in terms of complexity analysis and by means of numerical experiments for multi-asset options for up to seven dimensions.

FFT-based methods have been successfully applied, initially by Bakshi and Chen (1997), Scott (1997) and Carr and Madan (1999), in computational finance. In particular, the method of Dempster and Hong (2000) employs the FFT to price

The authors would like to thank Roger Lord, Fang Fang and Hisham bin Zubair for valuable discussions and Kees Lemmens for his assistance with the parallel code. The first author wishes to express his gratitude to the Dutch Technology Foundation STW for financial support.

options on more than one asset. Another similar approach for pricing coupon-bond options and swaptions has been presented by Singleton and Umantsev (2002). Here, we discuss a generalization of the convolution method by Lord *et al* (2008), which has been developed in particular for pricing early-exercise options. This pricing method is applicable to a wide variety of payouts and only requires knowledge of the characteristic function of the model. As such, the method is applicable within many regular affine models, among which is the class of exponential Lévy models.

This paper is organized into five sections. In Section 2, we describe the Fourier-based method for multi-asset options on tensor-product grids, called the multi-dimensional convolution method. Section 3 discusses the partitioning and corresponding parallelization of the method. This section also contains some parallel multi-asset results. In Section 4, we combine the method with a parallel sparse grid technique from Zenger (1990) and present results from numerical experiments on sparse grids. In Section 5, we draw our conclusions.

The options considered in this paper have their payout given by:

- $\left(\prod_{j=1}^{d} S_j^{1/d} - K\right)^+$ (call on the geometric average of the assets);
- $\left(\sum_{j=1}^{d} c_j S_j - K\right)^+$, with c_j the basket weights (basket call);
- options on the minimum or maximum of the underlying assets, for example $(K - \max_j S_j)^+$, put on the maximum of the underlying assets.

2 THE MULTI-DIMENSIONAL CONVOLUTION METHOD

The method presented falls into the category of transform methods. These methods are based on the risk-neutral valuation formula, which reads, for options on a single asset, as:

$$V(t, S(t)) = \mathrm{e}^{-r(T-t)}\mathbb{E}[V(T, S(T)) \mid S(t)] \tag{1}$$

Here V denotes the value of the option, r is the risk-free interest rate, t is the current time, T is the maturity date and S represents the price of the underlying. The interest rate, r, is assumed to be deterministic here. Equation (1) is an expectation and can be valuated directly, using numerical integration, provided that the probability density function is known. It can be written as:

$$V(t, x(t)) = \mathrm{e}^{-r(T-t)} \int_{K^*}^{\infty} (\mathrm{e}^{x(T)} - \mathrm{e}^{K^*}) f(x)\, \mathrm{d}x \tag{2}$$

with $K^* = \ln K$, $x(t) = \ln S(t)$ and $f(x)$ the probability density function. The value of $V(t, x(t))$ tends to $S(t)$ as K^* tends to $-\infty$ and hence the call price is not square integrable. Therefore, the payout is typically multiplied by a damping factor $\exp(\alpha K^*)$, with $\alpha > 0$, and the computation of the Fourier transform of the option value, ψ, can be performed by using the characteristic function, $\phi(\omega)$ (Carr and Madan (1999)):

$$\begin{aligned}\psi(\omega) &= \mathrm{e}^{-r(T-t)} \int_{-\infty}^{\infty} \mathrm{e}^{\mathrm{i}\omega K^*} \int_{K^*}^{\infty} \mathrm{e}^{\alpha K^*} (\mathrm{e}^{x} - \mathrm{e}^{K^*}) f(x)\, \mathrm{d}x\, \mathrm{d}K^* \\ &= \frac{\mathrm{e}^{-r(T-t)} \phi(\omega - (\alpha+1)\mathrm{i})}{\alpha^2 + \alpha - v^2 + \mathrm{i}(2\alpha+1)v}\end{aligned} \tag{3}$$

where the characteristic function of the underlying is defined by:

$$\phi(u) = \mathbb{E}[\mathrm{e}^{\mathrm{i}u \ln S(T)}] \tag{4}$$

To compute the call price, the inverse Fourier transform has to be computed:

$$V(t, x(t)) = \frac{\mathrm{e}^{-\alpha K^*}}{2\pi} \int_{-\infty}^{\infty} \mathrm{e}^{-\mathrm{i}\omega K^*} \psi(\omega)\, \mathrm{d}\omega \tag{5}$$

Following the same steps for basket options, however, would require the characteristic function of a log-basket value, which is not known in general. For common baskets, quite accurate approximations can be obtained by assuming that the basket itself is an asset following the same distribution as one of the underlying assets (Gentle (1993)). Here, we discuss a method which does not rely on such an approximation.

Like all transform-based methods, the convolution method (Lord *et al* (2008)) is based on the risk-neutral valuation formula (1). In the multi-dimensional version we need to compute:

$$V(t, \mathbf{x}(t)) = \mathrm{e}^{-r(T-t)} \int_{\mathbb{R}^d} V(T, \mathbf{y}) f(\mathbf{y} \mid \mathbf{x})\, \mathrm{d}\mathbf{y} \tag{6}$$

where $\mathbf{x} = \ln \mathbf{S}(t)$ is a vector of the log-asset prices, $\mathbf{y} = \ln \mathbf{S}(T)$ and $f(\mathbf{y} \mid \mathbf{x})$ is the probability density function of the transition of $\mathbf{x}$ at time t to $\mathbf{y}$ at time T.

REMARK 1 (Feynman–Kac theorem) For pricing multi-asset options the partial differential equation (PDE) approach, ie, solving the multi-dimensional Black–Scholes equation, is commonly used in the literature (Leentvaar and Oosterlee (2008); Tavella and Randall (2000)). This approach is connected to the risk-neutral expectation valuation by means of the Feynman–Kac theorem as follows. Suppose that we are given the system of stochastic differential equations:

$$\mathrm{d}S_i(t) = r S_i(t)\, \mathrm{d}t + \sigma_i S_i\, \mathrm{d}W_i(t), \quad i = 1, \ldots, d$$

with $\mathbb{E}\{\mathrm{d}W_i(t)\, \mathrm{d}W_j(t)\} = \rho_{ij}\, \mathrm{d}t$ and an option, V, such that:

$$V(t, \mathbf{S}(t)) = \mathrm{e}^{-r(T-t)} \mathbb{E}\{V(T, \mathbf{S}(T)) \mid \mathbf{S}(t)\}$$

with the sum of the first derivatives of the option square integrable. Then, the value, $V(t, \mathbf{S}(t))$, is the unique solution of the final condition problem:

$$\begin{cases} \dfrac{\partial V}{\partial t} + \dfrac{1}{2} \displaystyle\sum_{i,j=1}^{d} \left[\sigma_i \sigma_j \rho_{i,j} S_i S_j \frac{\partial^2 V}{\partial S_i \partial S_j} \right] + \sum_{i=1}^{d} \left[r S_i \frac{\partial V}{\partial S_i} \right] - r V = 0 \\ V(\mathbf{S}, T) = \text{given} \end{cases}$$

The solution in integral-form of this PDE can be obtained by use of a Green's function. The required characteristic function of the multi-dimensional convolution method is related to the Fourier transform of this Green's function. As a Fourier

transform of the Green's function will only be known for constant coefficient PDEs, this is also the setting for the multi-dimensional convolution method. Garroni and Menaldi (1992) provide a good description of Green's functions in the context of PDEs in finance.

The main premise of the multi-dimensional convolution method is that the transition density function $f(\mathbf{y} \mid \mathbf{x})$ is equal to the density of the difference of $\mathbf{y}$ and $\mathbf{x}$:

$$f(\mathbf{y} \mid \mathbf{x}) = f(\mathbf{y} - \mathbf{x}) \tag{7}$$

This holds for several models, such as geometric Brownian motion and, more generally, Lévy processes, which have independent increments. Then with $\mathbf{z} = \mathbf{y} - \mathbf{x}$, we have:

$$V(t, \mathbf{x}) = \mathrm{e}^{-r(T-t)} \int_{\mathbb{R}^d} V(T, \mathbf{x} + \mathbf{z}) f(\mathbf{z}) \, \mathrm{d}\mathbf{z} \tag{8}$$

which is a cross-correlation between V and f. The cross-correlation operator can be treated as a convolution operator (Gray and Goodman (1995)) and, therefore, the method is called the convolution method by Lord *et al* (2008).

The numerical valuation of (8) can be performed immediately for known probability density functions. For several asset price models, including the Lévy processes, however, only the characteristic functions are known. When transforming the cross-correlation operator into Fourier space, we have to deal with a product of the Fourier transform of the payout and the Fourier transform of the probability density function, which is the characteristic function.

The (continuous) Fourier transform of a function, however, can only be taken if the function is $\mathbf{L}^1$-integrable. This is typically not the case for multi-asset payout functions but damping techniques (Carr and Madan (1999) and Lord *et al* (2008)) are not available for multi-asset options in general. As an example we try to integrate a payout of a two-dimensional basket call, which is damped by $\mathrm{e}^{\alpha_1 x_1 + \alpha_2 x_2}$. The integral is split into two parts in the x_2-direction, one in which the payout function is only non-zero in a part of the x_1-direction and another in which x_1 can take all values:

$$\begin{aligned}
&\int_{\mathbb{R}^2} \mathrm{e}^{\alpha_1 x_1 + \alpha_2 x_2} K (c_1 \, \mathrm{e}^{x_1} + c_2 \, \mathrm{e}^{x_2} - 1)^+ \, \mathrm{d}x_1 \, \mathrm{d}x_2 \\
&\quad = K \int_{-\infty}^{-\ln c_2} \int_{\ln(1 - c_2 \, \mathrm{e}^{x_2}) - \ln c_1}^{\infty} \mathrm{e}^{\alpha_1 x_1 + \alpha_2 x_2} (c_1 \, \mathrm{e}^{x_1} + c_2 \, \mathrm{e}^{x_2} - 1) \, \mathrm{d}x_1 \, \mathrm{d}x_2 \\
&\qquad + K \int_{-\ln c_2}^{\infty} \int_{-\infty}^{\infty} \mathrm{e}^{\alpha_1 x_1 + \alpha_2 x_2} (c_1 \, \mathrm{e}^{x_1} + c_2 \, \mathrm{e}^{x_2} - 1) \, \mathrm{d}x_1 \, \mathrm{d}x_2
\end{aligned} \tag{9}$$

The second term in (9) is unbounded because of the integration over $\mathbb{R}$ for x_1. It is not possible to find a proper value for α_1 to bound this integral.

Instead of performing the Fourier transform analytically, however, in the multi-dimensional convolution method the computation is performed numerically and therefore the domain of integration has to be truncated. Since the density in (8)

decays to zero rapidly as $\mathbf{z} \to \pm\infty$, we truncate the infinite integration range without losing significant accuracy to $[\log S_0/K - L_i, \log S_0/K + L_i] \subset \mathbb{R}^d$, $i = 1, \ldots, d$. The direct construction of the discretized multi-dimensional convolution formula, which we perform in the following, via a Fourier series expansion of the continuation value replaces L^1-integrability on $(-\infty, \infty)$ with L^1-summability on a truncated computational domain, so that the restriction on α is removed. Experiments for many options contracts in Lord *et al* (2008) showed that one can choose $\alpha = 0$ in the convolution method and still obtain accurate option values. The discrete version will, however, resemble its continuous counterpart increasingly as the domain size increases.

So, our point of departure here will be a truncated domain, Ω_d. The size of Ω_d is chosen such that the error made due to truncation is negligible compared with the discretization error. To show the accuracy of the truncation, we refer the reader to Appendix A, Table A.1. In that table a numerical experiment with a plain vanilla call option is included, illustrating the effect of the truncation. It is shown numerically that a domain of size $\Omega_i = [\log S_0/K - L_i, \log S_0/K + L_i]$ with $L_i = 20\sigma_i$ gives highly accurate results, with σ_i denoting the standard deviation of the density. This domain size is set in all experiments to follow.

We now take a Fourier transform of (8) on the truncated domain Ω_d:

$$\begin{aligned} \mathrm{e}^{r(T-t)}\mathcal{F}\{V(t,\mathbf{x})\}(\boldsymbol{\omega}) &= \int_{\Omega_d} \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\mathbf{x}} \left[\int_{\Omega_d} V(T, \mathbf{x}+\mathbf{z}) f(\mathbf{z})\, \mathrm{d}\mathbf{z} \right] \mathrm{d}\mathbf{x} \\ &= \int_{\Omega_d}\int_{\Omega_d} \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}(\mathbf{x}+\mathbf{z})} V(T, \mathbf{x}+\mathbf{z}) f(\mathbf{z}) \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}\mathbf{z}}\, \mathrm{d}\mathbf{z}\, \mathrm{d}\mathbf{x} \end{aligned} \tag{10}$$

Here, the multi-dimensional Fourier transform, on the truncated domain, and its inverse are defined as:

$$\mathcal{F}\{h(\mathbf{x})\}(\boldsymbol{\omega}) = \int_{\Omega_d} \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\mathbf{x}} h(\mathbf{x})\, \mathrm{d}\mathbf{x},$$

$$\mathcal{F}^{\mathrm{inv}}\{H(\boldsymbol{\omega})\}(\mathbf{x}) = \frac{1}{(2\pi)^d} \int_{\Omega_d} \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}\mathbf{x}} H(\boldsymbol{\omega})\, \mathrm{d}\boldsymbol{\omega}$$

with, for example:

$$\begin{aligned} &\int_{\Omega_d} \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\mathbf{x}} h(\mathbf{x})\, \mathrm{d}\mathbf{x} \\ &\quad = \int_{-L_n}^{L_n} \cdots \int_{-L_1}^{L_1} \mathrm{e}^{\mathrm{i}\omega_1 x_1} \cdots \mathrm{e}^{\mathrm{i}\omega_d x_d} h(x_1, \ldots, x_d)\, \mathrm{d}x_1 \cdots \mathrm{d}x_d \end{aligned}$$

Changing the order of integration in (10) and using $\mathbf{y} = \mathbf{x} + \mathbf{z}$, we find:

$$\begin{aligned} \mathrm{e}^{r(T-t)}\mathcal{F}\{V(t,\mathbf{x})\}(\boldsymbol{\omega}) &= \int_{\Omega_d}\int_{\Omega_d} \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\mathbf{y}} V(T,\mathbf{y}) f(\mathbf{z})\, \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}\mathbf{z}}\, \mathrm{d}\mathbf{y}\, \mathrm{d}\mathbf{z} \\ &= \int_{\Omega_d} \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\mathbf{y}} V(T,\mathbf{y})\, \mathrm{d}\mathbf{y} \int_{\Omega_d} \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}\mathbf{z}} f(\mathbf{z})\, \mathrm{d}\mathbf{z} \\ &\approx \mathcal{F}\{V(T,\mathbf{y})\}(\boldsymbol{\omega})\phi(-\boldsymbol{\omega}) \end{aligned} \tag{11}$$

with the characteristic function, ϕ, defined by $\phi \equiv \int_{\mathbb{R}^d} \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}\mathbf{z}} f(\mathbf{z})\,\mathrm{d}\mathbf{z}$. After taking the inverse Fourier transform, the option price can be approximated by:

$$V(t,\mathbf{x}) \approx \mathrm{e}^{-r(T-t)}\mathcal{F}^{\mathrm{inv}}\{\mathcal{F}\{V(T,\mathbf{y})\}\phi(-\boldsymbol{\omega})\} \tag{12}$$

In this paper the asset prices are modeled as correlated log-normal distributions. The characteristic function can be computed via the probability density function:

$$f(\mathbf{x}) = \frac{1}{\sqrt{(2\pi)^d \mid \boldsymbol{\Sigma}\mid}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right)$$

where $\mu_j = (r - \delta_j - \frac{1}{2}\sigma_j^2)(T-t)$, $\boldsymbol{\Sigma}$ is the correlation matrix with $[\sigma]_{jk} = \rho_{jk}\sigma_j\sigma_k(T-t)$ and $|\boldsymbol{\Sigma}|$ its determinant. Dividend yields δ_j, volatilities σ_j and the correlation coefficients ρ_{jk} are assumed to be constant (as taking the Fourier transform makes sense for problems with constant coefficients). The characteristic function reads, using (4):

$$\phi(\boldsymbol{\omega}) = \exp\left(\mathrm{i}\sum_{k=1}^{d}\mu_k\omega_k - (T-t)\sum_{j=1}^{d}\sum_{k=1}^{d}\rho_{jk}\sigma_j\sigma_k\omega_j\omega_k\right) \tag{13}$$

REMARK 2 (Aliasing) Aliasing, a commonly observed feature when dealing with a convolution of sampled signals by means of the FFT, is not a problem in our application, as we encounter a convolution of a characteristic function and the discrete Fourier transform (DFT) of a vector with option values. The DFT is periodical but this would make the convolution circular only if the characteristic function would also be obtained by a DFT. However, we work with the analytical characteristic function, which is not periodic.

With the right to exercise at certain times, t_n, before the maturity date, T, the Bermudan option price is defined by:

$$V(t_n,\mathbf{x}(t_n)) \equiv \max\left\{E(t_n,\mathbf{x}(t_n)),\, \mathrm{e}^{-r(t_{n+1}-t_n)}\int_{\mathbb{R}^d} V(t_{n+1},\mathbf{y})\,f(\mathbf{y}\mid\mathbf{x})\,\mathrm{d}\mathbf{y}\right\} \tag{14}$$

with $E(t_n,\mathbf{x}(t_n))$ the exercise payout at t_n.

At each exercise date, t_n, the valuation of (14) can be interpreted as the computation of a European-style contract with maturity time t_{n+1} and "initial" time t_n, leading to the second term in the max-operator (which is compared with the payout). For the derivation of the multi-asset convolution method, we can therefore focus on (6) since this is also the major portion of computation in (14).

2.1 Discretization

Equation (12) can now be solved numerically with the help of multi-dimensional quadrature rules. A computation using the FFT requires equidistant grids for $\mathbf{x}$, $\mathbf{y}$

and $\boldsymbol{\omega}$:

$$x_{j,k_j} = x_{j,0} + k_j \, \mathrm{d}x_j \tag{15}$$

$$y_{j,k_j} = y_{j,0} + k_j \, \mathrm{d}y_j \tag{16}$$

$$\omega_{j,n_j} = \omega_{j,0} + n_j \, \mathrm{d}\omega_j \tag{17}$$

where the index j denotes the jth coordinate, $j = 1, \ldots, d$, and $k_j \in [0, N_j - 1]$ and $\mathrm{d}x_j = \mathrm{d}y_j$. The Nyquist relation has to be fulfilled to avoid aliasing:

$$\mathrm{d}x_j \cdot \mathrm{d}\omega_j = \frac{2\pi}{N_j} \tag{18}$$

We use the notation $x_{j,k_j} = x_{k_j}$, and denote the subscripts $k_1, \ldots, k_d$ by $\mathbf{k}$, the transformed vector by:

$$\mathcal{F}\{V(T, y_{k_1}, \ldots, y_{k_d})\} = \widehat{V}(T, \omega_{n_1}, \ldots, \omega_{n_d}) = \widehat{V}_{\mathbf{n}}$$

and the repeated sum by:

$$\sum_{k_1=0}^{N_1-1} \cdots \sum_{k_d=0}^{N_d-1} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{N-1}}$$

Approximating the inner integral of (12) by the trapezoidal rule, with the usual second-order accuracy, gives:

$$\begin{aligned} \widehat{V}_{\mathbf{n}} &:= \int_{\Omega_d} V(T, \mathbf{y}) \, \mathrm{e}^{\mathrm{i}\boldsymbol{\omega}\mathbf{y}} \, \mathrm{d}\mathbf{y} \\ &= \mathrm{d}\mathbf{Y} \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{N-1}} Z_{\mathbf{k}} V_{\mathbf{k}} \exp(\mathrm{i}\omega_{n_1} y_{k_1} + \cdots + \mathrm{i}\omega_{n_d} y_{k_d}) + \mathbf{O}\left(\sum_{j=1}^{d} \mathrm{d}y_j^2\right) \end{aligned} \tag{19}$$

with $\mathrm{d}\mathbf{Y} = \prod_{j=1}^{d} \mathrm{d}y_j$, $Z_{\mathbf{k}} = \prod_{j=1}^{d} R_j(k_j)$ and the trapezoidal weights:

$$R_j(k_j) = \begin{cases} \dfrac{1}{2} & k_j = 0 \vee k_j = N_j - 1 \\ 1 & \text{otherwise} \end{cases}$$

The terms $\exp(\mathrm{i}\omega_{n_j} y_{k_j})$, $j = 1 \ldots d$, can be rewritten as:

$$\begin{aligned} \exp(\mathrm{i}\omega_{n_j} y_{k_j}) &= \exp(\mathrm{i}(\omega_{j,0} + n_j \, \mathrm{d}\omega_j)(y_{j,0} + k_j \, \mathrm{d}y_j)) \\ &= \exp(\mathrm{i}\omega_{j,0} y_{j,0}) \exp(\mathrm{i}(\omega_{j,0} k_j \, \mathrm{d}y_j + y_{j,0} n_j \, \mathrm{d}\omega_j) \exp\left(\frac{2\pi \mathrm{i} n_j k_j}{N_j}\right) \end{aligned}$$

We choose $\omega_{j,0} = -\frac{1}{2} N_j \, \mathrm{d}\omega_j$ and $y_{j,0} = -\frac{1}{2} N_j \, \mathrm{d}y_j$, meaning that the grids are centered at the origin. Furthermore, we introduce the standard DFT notation,

$W_{N_j} = \exp(-2\pi \mathrm{i}/N_j)$, and have:

$$\begin{aligned}\exp(\mathrm{i}\omega_{n_j} y_{k_j}) &= \exp(\mathrm{i}\omega_{j,0} y_{j,0}) \exp(\mathrm{i}(\omega_{j,0} k_j \, \mathrm{d}y_j + y_{j,0} n_j \, \mathrm{d}\omega_j)) W_{N_j}^{-n_j k_j} \\ &= \exp\left(\frac{2\pi \mathrm{i} N_j}{4}\right) \exp(-\pi \mathrm{i} k_j) \exp(-\pi \mathrm{i} n_j) W_{N_j}^{-n_j k_j} \\ &= (-1)^{k_j} (-1)^{n_j + N_j/2} W_{N_j}^{-n_j k_j}\end{aligned}$$

Equation (19) can be written as:

$$\widehat{V}_{\mathbf{n}} \approx \mathrm{d}\mathbf{Y} \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} G_{\mathbf{k}} V_{\mathbf{k}} \prod_{j=1}^{d} (-1)^{n_j + N_j/2} W_{N_j}^{-n_j k_j} \tag{20}$$

with $G_{\mathbf{k}} = Z_{\mathbf{k}} \prod_{j=1}^{d} (-1)^{k_j}$. Recognizing that:

$$\mathcal{D}_d\{f_{\mathbf{k}}\} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} f_{\mathbf{k}} \prod_{j=1}^{d} \mathrm{e}^{2\pi \mathrm{i} n_j k_j / N_j} = \sum_{\mathbf{k}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} f_{\mathbf{k}} W_{\mathbf{N}}^{-\mathbf{nk}}$$

$$\mathcal{D}_d^{\mathrm{inv}}\{F_{\mathbf{n}}\} = \frac{1}{\prod_{j=1}^{d} N_j} \sum_{\mathbf{n}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} F_{\mathbf{n}} \prod_{j=1}^{d} \mathrm{e}^{-2\pi \mathrm{i} n_j k_j / N_j} = \frac{1}{\prod_{j=1}^{d} N_j} \sum_{\mathbf{n}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} F_{\mathbf{n}} W_{\mathbf{N}}^{\mathbf{nk}}$$

are the DFT and inverse Fourier transform, respectively, we have:

$$\widehat{V}_{\mathbf{n}} \approx \mathrm{d}\mathbf{Y} \prod_{j=1}^{d} ((-1)^{n_j + N_j/2}) \cdot \mathcal{D}_d[G_{\mathbf{k}} V_{\mathbf{k}}] \tag{21}$$

where $\mathcal{D}_d$ is the d-dimensional (or d-times repeated) DFT. The outer integral of (12) is treated by the left-hand rectangle rule in accordance with the error analysis in Lord *et al* (2008). So, we have:

$$\begin{aligned}V(t, \mathbf{x_m}) &= \mathrm{e}^{-r(T-t)} \mathcal{F}^{\mathrm{inv}}(\widehat{V}_{\mathbf{n}} \cdot \phi_{\mathbf{n}}) \approx \frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d} \int_{\Omega_d} \widehat{V}_{\mathbf{n}} \phi_{\mathbf{n}} \, \mathrm{e}^{-\mathrm{i}\boldsymbol{\omega}\mathbf{x}} \, \mathrm{d}\boldsymbol{\omega} \\ &= \mathrm{d}\boldsymbol{\Omega} \frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d} \sum_{\mathbf{n}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} \widehat{V}_{\mathbf{n}} \phi_{\mathbf{n}} \, \mathrm{e}^{-\mathrm{i}\omega_{n_1} x_{m_1} - \cdots - \mathrm{i}\omega_{n_d} x_{m_d}}\end{aligned} \tag{22}$$

with $\phi_{\mathbf{n}} = \phi(-\omega_{n_1}, \ldots, -\omega_{n_d})$, and by using (18):

$$\mathrm{d}\boldsymbol{\Omega} = \prod_{j=1}^{d} \mathrm{d}\omega_j = \prod_{j=1}^{d} \frac{2\pi}{N_j \, \mathrm{d}y_j} = \frac{(2\pi)^d}{N^d \, \mathrm{d}\mathbf{Y}} \tag{23}$$

Again, we can replace:

$$\mathrm{e}^{-\mathrm{i}\omega_{n_j} x_{m_j}} = (-1)^{m_j} (-1)^{n_j + N_j/2} W_{N_j}^{n_j m_j} \tag{24}$$

Combining (22), (23) and (24) gives:

$$V(t, \mathbf{x_m}) \approx \frac{\mathrm{e}^{-r(T-t)}}{N^d \, \mathrm{d}\mathbf{Y}} \sum_{\mathbf{n}=\mathbf{0}}^{\mathbf{N}-\mathbf{1}} \widehat{V}_{\mathbf{n}} \cdot \phi_{\mathbf{n}} \prod_{j=1}^{d} (-1)^{m_j} (-1)^{n_j + N_j/2} W_{N_j}^{n_j m_j} \tag{25}$$

We see that the products $(-1)^{n_j+N_j/2}$ vanish when combining (25) and (20). The combination of the two discretized integrals leads to the discrete multi-dimensional convolution method:

$$V(t, \mathbf{x_m}) = \frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d} \prod_{j=1}^{d} (-1)^{m_j} \mathcal{D}_d^{\mathrm{inv}} [\phi_{\mathbf{n}} \mathcal{D}_d \{V_{\mathbf{k}} G_{\mathbf{k}}\}] \tag{26}$$

3 PARALLEL PARTITIONING

Equation (26) is set up for tensor-product multi-dimensional grids. For an increasing number of dimensions, however, the total number of points will increase exponentially. This so-called curse of dimensionality (Bellman (1961)) renders many sequential algorithms useless. Even on state-of-the-art sequential computers, the memory is not large enough to store vectors of size $N = \prod_{j=1}^{d} N_j$, for d sufficiently large. One of the possibilities to solve on finer grids is to partition the problem and solve the different parts in parallel.

The partitioning chosen here is based on the down-sampling method (Gray and Goodman (1995)). The basis of this method is a splitting of the input vector (in our case the Fourier transformed payout) into two parts: one part containing the even and one containing the odd points. These two DFTs can be computed independently and their result can be added. A straightforward sequential implementation of the DFT of size N uses N^2 computations, whereas the partitioned version needs $(N/2)^2$ computations for each DFT plus one summation. This partitioning can be continued. It will converge to $O(N \log N)$ computations, in the same manner as the FFT. Taking N as a power of two is the most efficient choice. Equation (26) is based on the transform of the payout and the inverse transform of the product. This structure complicates the partitioning to some extent, but it can still be used, as described in the following.

Consider first the one-dimensional version of (26):

$$H_m = \sum_{n=0}^{N-1} \phi_n \widehat{V}_n W_N^{mn} \tag{27}$$

where we omit the discounting factor. As mentioned, we split (27) into two sums of size $M = N/2$:

$$\begin{aligned} H_m &= \sum_{n=0}^{M-1} \phi_{2n} \widehat{V}_{2n} W_N^{2nm} + \sum_{n=0}^{M-1} \phi_{2n+1} \widehat{V}_{2n+1} W_N^{(2n+1)m} \\ &= \sum_{n=0}^{M-1} \phi_{2n} \widehat{V}_{2n} W_M^{nm} + W_N^m \sum_{n=0}^{M-1} \phi_{2n+1} \widehat{V}_{2n+1} W_M^{nm} \end{aligned} \tag{28}$$

where we use:

$$W_N^{2nm} = \mathrm{e}^{-2\pi \mathrm{i} 2nm/N} = \mathrm{e}^{-2\pi \mathrm{i} mn/M} = W_M^{mn}$$

The two DFTs in (28) can be solved in parallel. The inner parts ϕ_{2n}, ϕ_{2n+1}, $\widehat{V}_{2n}$ and $\widehat{V}_{2n+1}$ can be computed by addressing the appropriate points. Here $\widehat{V}_{2n}$ and $\widehat{V}_{2n+1}$ are based on a size N vector, $\widehat{V}$, which we partition as well.

For the even elements, we can write:

$$\begin{aligned}
\widehat{V}_{2n} &= \sum_{k=0}^{N-1} G_k V_k W_N^{2nk}, \\
&= \sum_{k=0}^{M-1} G_k V_k W_M^{nk} + \sum_{k=M}^{N-1} G_k V_k W_M^{nk} \\
&= \sum_{k=0}^{M-1} G_k V_k W_M^{nk} + W_M^{nM} \sum_{k=0}^{M-1} G_{k+M} V_{k+M} W_M^{nk}
\end{aligned} \tag{29}$$

and for the odd elements:

$$\widehat{V}_{2n+1} = \sum_{k=0}^{M-1} G_k V_k W_N^k W_M^{nk} + W_N^M W_M^{nM} \sum_{k=0}^{M-1} G_{k+M} V_{k+M} W_N^k W_M^{nk} \tag{30}$$

We observe that (29) and (30) are again sums of two DFTs of size $N/2$. When the splittings (28), (29) and (30) are combined, we find the one-dimensional partitioned version of (27).

The multiple partitioned version is based on a repetition of this splitting which we derive for β parts, with β a power of two. The size of the computations is then $M = \beta^{-1} N$. The points used in the splitting of the inverse transform are given by $\beta n + q$, with $q \in [0, \beta - 1]$. So, the multiple partitioned version of (27) reads, using $W_N^{\beta nm} = W_M^{nm}$:

$$H_m = \sum_{q=0}^{\beta-1} \sum_{n=0}^{M-1} \phi_{\beta n+q} \widehat{V}_{\beta n+q} W_N^{m(\beta n+q)} = \sum_{q=0}^{\beta-1} W_N^{mq} \sum_{n=0}^{M-1} \phi_{\beta n+q} \widehat{V}_{\beta n+q} W_M^{mn} \tag{31}$$

The partitioning into the odd and even parts can now be included:

$$\begin{aligned}
\widehat{V}_{\beta n+q} &= \sum_{k=0}^{N-1} V_k G_k W_N^{-(\beta n+q)k} \\
&= \sum_{p=0}^{\beta-1} \sum_{k=0}^{M-1} V_{k+pM} G_{k+pM} W_N^{-(\beta n+q)(k+pM)} \\
&= \sum_{p=0}^{\beta-1} W_\beta^{-pq} \sum_{k=0}^{M-1} V_{k+pM} G_{k+pM} W_M^{-nk} W_N^{-qk}
\end{aligned} \tag{32}$$

where we used:

$$W_N^{-\beta npM} = \mathrm{e}^{2\pi inp} = 1 \quad \text{and} \quad W_N^{-pqM} = W_\beta^{-pq}$$

Combining (31) and (32), we obtain the one-dimensional multiple split version of (27):

$$\begin{aligned} H_m &= \sum_{q=0}^{\beta-1} W_N^{mq} \sum_{n=0}^{M-1} \phi_{\beta n+q} W_M^{mn} \sum_{p=0}^{\beta-1} W_\beta^{-pq} \sum_{k=0}^{M-1} V_{k+pM} G_{k+pM} W_M^{-nk} W_N^{-qk} \\ &= \sum_{p=0}^{\beta-1} \sum_{q=0}^{\beta-1} W_\beta^{-pq} W_N^{mq} \mathcal{D}^{\mathrm{inv}}(\phi_{\beta n+q} \mathcal{D}[V_{k+pM} G_{k+pM} W_N^{-qk}]) \end{aligned} \tag{33}$$

This partitioning can be generalized to the multi-dimensional case. We then have a partitioning vector $\boldsymbol{\beta}$ containing β_j parts for each coordinate j. The points in the outer transform (31) are represented by $\boldsymbol{\beta n} + \boldsymbol{q} = (\beta_1 n_1 + q_1, \ldots, \beta_d n_d + q_d)$. The points of the payout addressed in (32) are represented by $\mathbf{k} + \mathbf{pM} = (k_1 + p_1 M_1, \ldots, k_d + p_d M_d)$. When using the multi-dimensional summation, the multiple partitioned version of (26) reads:

$$V(t, \mathbf{x}) = \frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d} \sum_{\mathbf{p}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} \sum_{\mathbf{q}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} W_{\boldsymbol{\beta}}^{-\mathbf{pq}} W_{\mathbf{N}}^{\mathbf{mq}} \mathcal{D}_d^{\mathrm{inv}}[\phi_{\boldsymbol{\beta}\mathbf{n}+\mathbf{q}} \mathcal{D}_d\{V_{\mathbf{k}+\mathbf{pM}} G_{\mathbf{k}+\mathbf{pM}} W_{\mathbf{N}}^{-\mathbf{qk}}\}] \tag{34}$$

We see that if one of the coordinate grids is split into two parts, ie, $\beta_k = 2$, $\beta_{j\neq k} = 1$, the computation of (34) would be a combination of four DFTs of size $(N_k/2) \prod_{j=1, j\neq k}^{d} N_j$. If the same coordinate would be partitioned again, we would deal with 16 DFTs of size $(N_k/4) \prod_{j=1, j\neq k}^{d} N_j$. The parallel efficiency is low in this case, as the number of processors needed grows quadratically with β_j. Therefore, we rewrite (34) as:

$$V(t, \mathbf{x}) = \frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d} \sum_{\mathbf{q}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} W_{\mathbf{N}}^{\mathbf{mq}} \mathcal{D}_d^{\mathrm{inv}} \left[\phi_{\boldsymbol{\beta}\mathbf{n}+\mathbf{q}} \mathcal{D}_d \left\{ \sum_{\mathbf{p}=\mathbf{0}}^{\boldsymbol{\beta}-\mathbf{1}} V_{\mathbf{k}+\mathbf{pM}} G_{\mathbf{k}+\mathbf{pM}} W_{\mathbf{N}}^{-\mathbf{qk}} W_{\boldsymbol{\beta}}^{-\mathbf{pq}} \right\} \right] \tag{35}$$

In this case the computations are partitioned over the $\mathbf{q}$ sum into $B = \prod_{j=1}^{d} \beta_j$ parts. Each processor now has to compute the $\mathbf{p}$ parts of the payout function. The summation over $\mathbf{p}$ could also be performed in parallel with communication among the processors. A drawback of allowing communication for high-dimensional problems is the need to transfer very large vectors from one processor to another. We certainly need the communication when solving early exercise options in parallel, but not for European options. So, early exercise options would be parallelized efficiently on a parallel machine with some form of shared memory. An alternative to this type of parallelization may be the sparse grid method for which parallelization is straightforward. We focus on the communicationless version of this parallel approach and solve European-style options with it.

3.1 Complexity analysis

We now evaluate the parallelization technique to solve (26) by a complexity analysis, and first summarize the procedure to solve (26).

1) Compute the payout on the tensor-product grid.
2) Multiply the payout by the function $G_{\mathbf{k}}$.
3) Take the FFT.
4) Multiply the result by the characteristic function.
5) Take the inverse FFT of the product.
6) Multiply it by the discount factor.
7) For Bermudan options: take the maximum of this value and the payout function at t_n. Repeat the procedure from step 2 until t_0 is reached.

The construction of the payout function is a significantly faster procedure than the computation of the two FFTs and the multiplication by the characteristic function. We distinguish three portions of time consumption during the solution process of European options:

- T_{pay} is the time needed to construct the payout including the multiplication with the function $Z_{\mathbf{k}}$ and $W_{\mathbf{N}}^{-\mathbf{qk}}$ in (35);
- T_{four} is the time in steps 3 to 6 in the algorithm,
- T_{add} is the additional time needed for starting the computation, reading and writing files.

We assume here that T_{add} is negligible. The total time needed to compute (26) is then $T_{\text{tot}} \approx T_{\text{pay}} + T_{\text{four}}$. We further assume that $T_{\text{four}} = AT_{\text{pay}}$. If technique (35) is used and we partition the problem into B parts, the computational time per processor is:

$$T_{\text{tot,split}} = T_{\text{pay}} + \frac{1}{B} T_{\text{four}} = \frac{A+B}{B} T_{\text{pay}} \tag{36}$$

with $B = \prod_{j=1}^{d} \beta_j$. If there are Q identical processors available, then the parts B can be distributed over the Q processors. Ideally Q is a divisor of B. The number of parallel processes is therefore equal to $\lceil B/Q \rceil$ and the computational time reads:

$$T_{\text{tot,split}} = \left\lceil \frac{B}{Q} \right\rceil \frac{A+B}{B} T_{\text{pay}} \tag{37}$$

So, when $B = Q$, we see that T_{pay} is not subdivided in our parallel approach, as explained above. In our applications, typically, $A \in [4, 12]$ for $B = 1$.

3.2 Full grid experiments

We evaluate the CPU times of the parallel multi-dimensional convolution method for some multi-dimensional experiments on tensor-product grids. We first evaluate the option on the geometric average for which we compare the numerical result with an analytic solution (Berridge and Schumacher (2004)). This solution can be

TABLE 1 Option prices for the four-dimensional geometric average call option with the parallel timings (in seconds), $Q = B$. The last column gives A (36).

$d = 4$	Call on the geometric average			CPU times with (35)				
n_f	Price	Error	Ratio	$B = 1$	$B = 2$	$B = 4$	$B = 16$	A
3	1.962	2.0×10^{-1}	5.3	<0.1	<0.1	<0.1	<0.1	
4	2.128	3.8×10^{-2}	5.4	<0.1	<0.1	<0.1	<0.1	
5	2.156	9.3×10^{-3}	4.1	0.5	0.2	0.1	<0.1	4.5
6	2.163	2.3×10^{-3}	4.0	9.4	4.9	3.0	1.6	6.2
7	2.165	5.8×10^{-4}	4.0	164.1	85.1	45.2	25.2	7.1

obtained by the use of a coordinate transformation $y = \prod_{j=1}^{d} \mathrm{e}^{x_j/d}$. The option price can then be obtained via the one-dimensional Black–Scholes formula with:

$$\widehat{\sigma} = \sqrt{\frac{1}{d^2} \sum_{j=1}^{d} \sum_{K=1}^{d} \rho_{jk} \sigma_j \sigma_k}, \quad \widehat{\delta} = \frac{1}{d} \sum_{j=1}^{d} \left(\delta_i + \frac{1}{2} \sigma_i^2 \right) - \frac{1}{2} \widehat{\sigma}^2$$

In Table 1, the prices for the four-dimensional call option on the geometric average of the assets are presented for a different number of grid points. The first column in Table 1 represents the number of grid points per coordinate $N_j = 2^{n_f}$, $j = 1, \ldots, 4$. The final computation, $n_f = 7$, requires 4 GB of memory and has a complexity of 2^{28} points.

The option parameters chosen are $r = 0.06$, $\sigma_j = 0.2$, $\delta_j = 0.04$, $\rho_{jk} = 0.25$ if $j \neq k$ and $T = 1$. The strike price is €40, as is $\mathbf{S}(0)$.

The desired accuracy of errors being less than €0.01 is achieved for $n_f = 5$. We observe a second-order convergence on the finer grids. The right-hand side of Table 1 presents timings on a parallel machine, which consists of nodes with two processors each, with 8 GB of memory. Parameter A, as in (36) is also given.

Based on these results we conclude that the partitioning strategy is useful in reducing the total CPU time. Parallel efficiency would improve on finer grids and in higher dimensions.

We now consider problems that require more than the maximum available physical memory per processor. The parallel partitioning is then mandatory. In Table 2, we present the prices of a digital put on the geometric average of five assets. As the payout function of the digital option (it will pay an amount of €1 when the geometric average is less than the strike price in our experiment) has a discontinuity along the hypersurface $\prod_{j=1}^{d} \mathrm{e}^{x_j/d} = 1$, it is expected that this leads to only first-order error convergence. Table 2 indeed displays first-order convergence (again the exact solution is known for the digital put on the geometric average) and we see that a grid size with $n_f = 6$ is not sufficient to reach the desired accuracy.

Table 3 then presents the solution of a six-dimensional standard basket put with equally weighted assets ($c_i = \frac{1}{6}$). The error convergence is irregular for this payout, but at least of second order. The size, $n_f = 5$, is again sufficient to reach the desired

TABLE 2 Option prices for the five-dimensional geometric average digital put, plus parallel timing results and parameter A from (36).

$d = 5$	Digital put on the geometric average			CPU times		
n_{f}	Price	Error	Ratio	$B = 4$	$B = 32$	A
2	0.81	3.36×10^{-1}	1.49	<0.1	<0.1	
3	0.32	1.49×10^{-1}	2.26	<0.1	<0.1	
4	0.40	7.43×10^{-2}	2.00	0.2	0.1	4.0
5	0.43	3.71×10^{-2}	2.00	1.8	1.1	4.5
6	0.45	1.86×10^{-2}	2.00	295.6	91.1	8.7

TABLE 3 Option prices for the six-dimensional basket put, plus parallel timing results.

$d = 6$	Basket put			CPU times	
n_{f}	Price	Error	Ratio	$B = 4$	$B = 32$
2	1.26	1.25		<0.01	<0.01
3	1.52	2.63×10^{-1}	4.7	0.09	<0.01
4	1.51	1.70×10^{-2}	15.5	5.02	1.2
5	1.50	2.62×10^{-3}	6.5	334.34	111.1

accuracy. The CPU times for $B = 32$ in Tables 2 and 3 are estimated times when the number of processors Q is equal to the number of parts B.

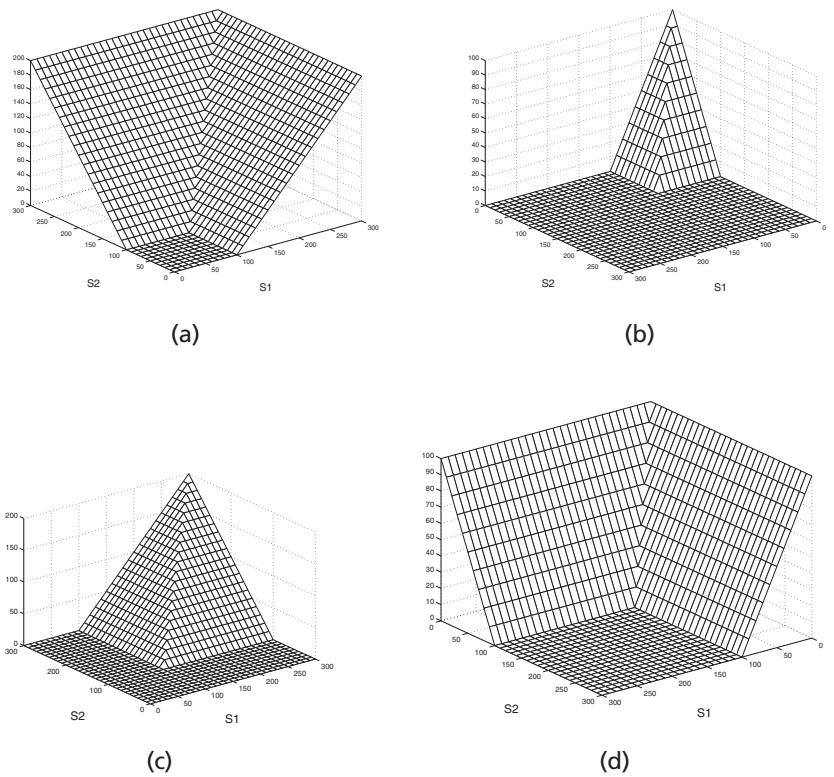
The hedge parameters can easily be obtained using the convolution method. We refer to Appendix B for the derivation and some numerical results for Delta.

In the next section, we describe the sparse grid technique that we use to solve multi-asset options. The sparse grid technique can be chosen if the required memory or the required number of processors for the partitioned full grid version is too large. However, the efficient use of the sparse grid technique in computational finance is seriously restricted by the types of multi-asset option contracts in use. An acceptable accuracy with the sparse grids method can be expected if the solution has bounded mixed derivatives. The payouts of the examples presented in Tables 1–3 do not have this property. It may be possible to transform a payout so that the kink (or discontinuity for a digital option) is aligned with a grid line (Leentvaar and Oosterlee (2008)), but this cannot be done for every payout. A call or put option based on the maximum or minimum of the underlying assets has its non-differentiability along grid lines, see Figure 1. It is therefore expected that these options can be handled rather well in the sparse grid setting.

4 SPARSE GRIDS

The partitioning of (26) to obtain (35) is not sufficient to deal with the curse of dimensionality. It helps to obtain problems of moderate size that can fit into the available memory or to speed up the computation of medium-sized problems.

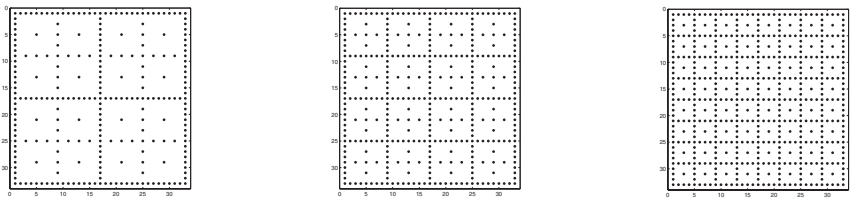
FIGURE 1 Payout for two-dimensional options on the maximum or minimum of assets: (a) call on maximum; (b) put on maximum; (c) call on minimum; (d) put on minimum.



A five-dimensional problem with 2^6 grid points per coordinate would need vectors containing 2^{30} elements (about 16 GB of memory). With a maximum of 4 GB of memory per processor, we could solve the problem on four processors (splitting the problem twice). However, a seven-dimensional problem with 2^7 grid points per coordinate is too large to partition with a moderate number of processors. Vectors on this grid will consist of 2^{49} elements. If processors with 4 GB memory are available, the number of splittings required is 21 and we would have 2^{21} subproblems of a reasonable size. Of course, several subproblems can be solved on a single processor as the FFT-based computation is typically fast. With 2^{11} subproblems on one processor, we would need $2^{10} \approx 1{,}000$ processors. The (computer-dependent) restrictions on the number of dimensions and the tensor-product grid sizes that can be efficiently handled in parallel can be calculated easily.

The sparse grid method, developed by Zenger and co-workers (Bungartz and Griebel (2004); Zenger (1990)), is an approach that allows the solution of problems

FIGURE 2 Construction of a two-dimensional sparse grid. Combined solution with different values of b: (a) $b = 1$; (b) $b = 2$; (c) $b = 3$.



(a) (b) (c)

with higher dimensionality. The solution is obtained via a combination of approximations obtained on relatively coarse high-dimensional grids, which are called subproblems here. The solution of this combination technique mimics the solution on a full tensor-product grid.

Let us consider a full grid with $N_j = 2^{n_s}$, ie, the number of grid points for coordinate direction j. The sparse grid method combines subproblems with a maximum number of 2^{n_s}-points in one coordinate. The other coordinates of the subproblem are discretized with a minimum number of grid points per direction, 2^b, called "the base" here. The number of subproblems to combine depends on the number of dimensions, the base and the number of points of the full grid to mimic. These subproblems are ordered in d layers with the complexity of each subproblem within a layer being approximately the same. The combination of the subproblem solutions, by means of interpolation, at a certain layer ℓ, is just the sum of these solutions on the grids. The first layer of subproblems consists of grids of size $(2^{n_s} \times 2^b \times \cdots \times 2^b)$. The layer number for this grid is $\ell = n_s + (d-1)b$. Grids of size $(2^b \times 2^b \times \cdots \times 2^{n_s})$ and $(2^{b+2} \times 2^b \times \cdots \times 2^{n_s-2})$ are, for example, also present in this layer. The next layer is of size $\ell = n_s - 1 + (d-1)b$. To form the solution a weighted combination of layers is taken with the binomial coefficients as the weights (Bungartz and Griebel (2004)):

$$V_{\text{combined}} = \sum_{j=1}^{d} (-1)^{j+1} \binom{d-1}{j-1} \sum_{q \in \mathcal{I}_{d,\ell}} V_q \tag{38}$$

where $\mathcal{I}_{d,\ell}$ represents the indices of all d-dimensional grids on layer ℓ and q is one of its elements. The combined solution leads to the sparse grid solution, see Figure 2.

EXAMPLE 1 Consider a three-dimensional grid of size 16^3, whose solution we would like to mimic using the sparse grid method. Now $n_s = 4$, $d = 3$ and we have three layers of subproblems. Suppose that we need a grid that consists of at least four points per coordinate ($b = 2$), then we find:

- first layer, $\ell = 8$, with grids $(16 \times 4 \times 4)$, $(8 \times 8 \times 4)$, $(4 \times 16 \times 4)$, $(8 \times 4 \times 8)$, $(4 \times 8 \times 8)$ and $(4 \times 4 \times 16)$;
- second layer, $\ell = 7$, with grids $(8 \times 4 \times 4)$, $(4 \times 8 \times 4)$ and $(4 \times 4 \times 8)$;
- third layer, $\ell = 6$, with grid $(4 \times 4 \times 4)$.

Although we do not have an exponentially growing number of points with this method, the number of subproblems to be solved, $\mathcal{M}_{d,\ell}$, increases significantly for increasing dimensions. For a given n_s, b and d it reads:

$$\mathcal{M}_{d,\ell} = \binom{\ell - d(b-1) - 1}{d-1} \tag{39}$$

The total number of points on this layer reads, $\mathcal{N}_\ell = 2^{\ell+(d-1)} \cdot \mathcal{M}_{d,\ell}$, and the total number of points in a sparse grid computation, mimicking a full grid of size 2^{n_s} in each direction, reads:

$$\mathcal{N}_\mathrm{total} = 2^{n_\mathrm{s}-d+1+(d-1)b} \sum_{j=1}^{d} 2^{d-j} \binom{n_\mathrm{s} - j - b + d}{d-1} \tag{40}$$

If, however, even a sparse grid subproblem does not fit into the memory, we additionally have to make use of the parallelization strategy from Section 3, partitioning the multi-dimensional convolution method for all of the subproblems in a sparse grid layer. Let us consider, as an example, a seven-dimensional problem with $n_\mathrm{s} = 10$ and $b = 2$. This problem requires 2^{28} grid points for the subproblems in the top layer. The total number of subproblems in that layer is 1,716, but these subproblems need to be partitioned once according to the splitting in Section 3. Therefore, we deal with 3,432 subproblems of roughly 2^{27} points. The full grid problem would require 2^{70} grid points (2^{43} GB), which is infeasible. The overall sparse grid complexity is 2^{39} points, subdivided into 5,147 subproblems.

The accuracy of the convolution method can help to gain some insight in the error for the sparse grid case. For a single-asset option with the asset modeled by geometric Brownian motion a second-order full grid convergence was derived by Lord *et al* (2008). We also assume an error of $\mathbf{O}(\Delta x^2)$ for the multi-dimensional problem, discretized on an equidistant grid with mesh size Δx. For the sparse grid integration technique, it was shown, for example, by Gerstner and Griebel (1998) that the order of convergence is $\mathbf{O}(\Delta x^2(\log(\Delta x^{-1}))^{d-1})$, with Δx the smallest mesh size occurring, for problems with bounded mixed derivatives. We assume here that this sparse grid error expansion is also valid for the contracts evaluated in the next section. A required sparse grid accuracy, connected to the maximum number of grid points in one direction $N_S = 2^{n_\mathrm{s}}$, can be related "globally" to the number of grid points, $N_j = 2^{n_\mathrm{f}}$, in the full grid case, as follows:

$$C_\mathrm{f} 2^{-2n_\mathrm{f}} = C_\mathrm{s} 2^{-2n_\mathrm{s}} n_\mathrm{s}^{d-1} \tag{41}$$

with constants C_f and C_s. The solution to this equation is given by:

$$n_\mathrm{s} = \exp\left(-\mathcal{L}\left(-\frac{\ln 4}{d-1}\exp(D)\right) - D\right)$$

$$D = \frac{-n_\mathrm{f}\ln 4 + \ln C_\mathrm{f} - \ln C_\mathrm{s}}{d-1}$$

and $\mathcal{L}$ is the Lambert W function.[1] With this expression, we can compute the required number of grid points for a sparse grid computation to mimic a certain full grid problem with grid size n_f, given the desired accuracy ε, constant C_f and number n_f and the upper-bound of C_s. The constants C_f and C_s can be determined from a small-sized experiment taking into account that, in particular, C_s is problem dependent.

4.1 Sparse grid computations

In the sparse grid setting, we can use the convolution algorithm as in the full grid case. In fact, the sparse grid method can be coded as an outer loop running over all subproblems. Within the loop, the convolution method is called with the desired grid parameters. We developed the algorithm so that if the subproblems in the sparse grid are additionally partitioned as described in Section 3, the number of parallel tasks increases to $U = \mathcal{N}B$, where B is the number of parts of a subproblem. The algorithm loops over all tasks U. Every task is sent to a different processor as soon as the processor is available. The maximum number of tasks in a problem is limited to 2^{31} on a 32-bit machine and 2^{63} on a 64-bit machine. As soon as a subproblem is solved, the solution (an option value) is returned to the master process and summed. After this task is performed, a new task can be assigned to this processor. This kind of parallel coding is not straightforward, due to the three different types of partitioning within the algorithm, but it can be used on a heterogeneous cluster.

We now perform numerical experiments with option contracts on the maximum or minimum of the underlying assets. The option parameters for these experiments are:

- $K = 100$, $T = 1$ year, $r = 0.045$;
- $\sigma_1 = 0.25$, $\sigma_2 = 0.35$, $\sigma_3 = 0.20$, $\sigma_4 = 0.25$, $\sigma_5 = 0.20$, $\sigma_6 = 0.21$ and $\sigma_7 = 0.27$;
- $\delta_1 = 0.05$, $\delta_2 = 0.07$, $\delta_3 = 0.04$, $\delta_4 = 0.06$, $\delta_5 = 0.04$, $\delta_6 = 0.03$ and $\delta_7 = 0.02$;
- $R = \begin{pmatrix} 1.00 & -0.65 & 0.25 & 0.20 & 0.25 & -0.05 & 0.05 \\ -0.65 & 1.00 & 0.50 & 0.10 & 0.25 & 0.11 & -0.016 \\ 0.25 & 0.50 & 1.00 & 0.37 & 0.25 & 0.21 & 0.076 \\ 0.20 & 0.10 & 0.37 & 1.00 & 0.25 & 0.27 & 0.13 \\ 0.25 & 0.25 & 0.25 & 0.25 & 1.00 & 0.14 & -0.04 \\ -0.05 & 0.11 & 0.21 & 0.27 & 0.14 & 1.00 & 0.19 \\ 0.05 & -0.016 & 0.076 & 0.13 & -0.04 & 0.19 & 1.00 \end{pmatrix}$

with R the matrix with the correlation coefficients ρ_{jk}.

[1] The Lambert W function is the solution of $\mathcal{L}(x)\,\mathrm{e}^{\mathcal{L}(x)} = x$.

TABLE 4 European and Bermudan four-dimensional put option on the maximum of the underlying assets on a full grid. The Bermudan contract has 10 exercise dates.

$d = 4$	European			Bermudan		
n_f	Price	Error	Ratio	Price	Error	Ratio
3	0.38	7.38×10^{-1}		0.61	5.06×10^{-1}	
4	0.87	2.51×10^{-1}	2.94	0.53	9.25×10^{-1}	0.55
5	1.05	6.82×10^{-2}	3.67	1.87	3.36×10^{-1}	2.75
6	1.10	1.76×10^{-2}	3.88	1.85	2.88×10^{-2}	11.67
7	1.11	4.51×10^{-3}	3.90	1.84	5.37×10^{-3}	5.36

We start with the four-dimensional problem and compare the sparse and the full grid results. The parameters for this experiment are listed above, where we take the first four subscript entries. In Table 4, the results for the full grid experiment are presented for a European and a Bermudan contract. The Bermudan contract has 10 exercise dates during the lifetime of the option contract. The results are presented for grids with $N_j = 2^{n_f}$, $j = 1, \ldots, d$, points. We see a smooth convergence for this type of European contract and an accuracy better than €0.01 when $n_f = 7$. For the Bermudan contract, the convergence ratio is less smooth, but the accuracy is again satisfactory.

REMARK 3 (Smooth fit principle) It is well-known that in the case of American options under Black–Scholes dynamics the derivative of the value function is continuous (smooth fit principle). However, this is not the case when pricing Bermudan options, for which the function V will have a discontinuous first derivative. Although at the final exercise time the location of this discontinuity is known, this is not the case at previous exercise times. This may hamper the numerical treatment of options in the present context.

REMARK 4 (American options) In order to price American options, based on the present approach for Bermudan options, there are basically two approaches. One can compute a Bermudan option with many exercise dates, and thus very small time steps, as an approximation of an American option, or one can apply a repeated Richardson extrapolation. These two approaches have been applied to the univariate case by Lord *et al* (2008), in which it was shown that the Richardson extrapolation was superior in terms of accuracy and CPU time. The convergence of the Bermudan approach with many exercise dates was only of first order, whereas the Richardson extrapolation was significantly better than that. However, Lord *et al* (2008) achieved a very regular convergence of the pricing for Bermudan options with the convolution method by shifting the grids, so that the option value where the continuation and the payout values coincide was placed at a grid point. This was the reason for the accurate American options prices of Lord *et al* (2008). In the multivariate case, however, it is no longer possible to place an "early -exercise line", or even a higher-dimensional entity, completely on a grid line. Therefore, we cannot

achieve a regular convergence for Bermudan options and extrapolation cannot be used to price American options. American multi-asset options are, however, still rare financial contracts.

Although the results are satisfactory in the four-dimensional case, the Bermudan option contract, for example, cannot be computed with $n_f = 7$ in higher-dimensional cases without any form of communication. In Section 4, we derived an expression to compute the number of grid points $N_s = 2^{n_s}$ needed for mimicking the solution with sparse grid, given the accuracy and the number of grid points $N_f = 2^{n_f}$ in the full grid. The number of grid points is a rough estimate and we assume the value of C_f as an upper bound for C_s. With a desired accuracy of approximately 10^{-3}, we would need $n_f = 8$ in the full grid case. From the results in Table 4, we have $C_f \approx 74$ with $n_f = 7$. Solving (41), the number of grid points for the sparse grid computation is $n_s = 13$, to have an accuracy of 10^{-3}. The choice of the base b in the sparse grid technique has a major influence on the complexity. The convolution method does not work if one of the coordinates is discretized in only two points, so $b \geq 2$. It is also reasonable for accuracy reasons to use a higher base (Leentvaar and Oosterlee (2008)), for example $b = 3$ which means at least eight points per coordinate. This, however, also has a significant impact on the costs of the method.

In Table 5, the results for the put option and the maximum of the underlying assets are presented for the four- and five-dimensional case and for European and Bermudan options based on a sparse grid technique with $b = 3$. In this table, the first column represents the mimic of the full grid. With $n_s = 13$, the desired accuracy of 10^{-3} is reached. The sparse grid method also converges to the same value as the full grid case (see Table 4), although the convergence tends to be of first order and irregular. For the five asset problem (right-hand side of Table 5), we see the same behavior of the four asset problem by means of accuracy and convergence. Finally, the Bermudan option contract also reaches the desired accuracy when $n_s = 13$.

REMARK 5 (Irregular sparse grid convergence) The sparse grid convergence in Table 5 is somewhat irregular. Option pricing problems are typically characterized by payout functions that do not have bounded mixed derivatives. The max option, however, has this feature only along its axes (see Figure 1). So, in principle, we would expect the asymptotic theoretical optimal sparse grid convergence of $O(h^2(\log h^{-1})^{d-1})$. However, for the multi-dimensional Poisson equation and a smooth solution, bin Zubair *et al* (2007) showed that the theoretical sparse grid convergence was only achieved on relatively fine grids. Here we have the same situation. For the four-dimensional case of Table 5 we observe significantly improved sparse grid convergence rates on finer grids: $n_s = 14$: the error is 5.0×10^{-4} with convergence ratio 2.43, for $n_s = 15$: we have error 1.9×10^{-4} and ratio 2.79, for $n_s = 16$, the error equals 6.0×10^{-5}, and ratio 3.1, while for $n_s = 17$ the error is 1.8×10^{-5} and the ratio is 3.36.

The interesting point is the CPU time. In Table 4, we have an accuracy of 4.5×10^{-3} when $n_f = 7$ for the full grid European four-dimensional option. The CPU time on 16 equivalent processors for the full grid problem (see Table 1) is 25 seconds. We have an accuracy of 4.68×10^{-3} with $n_s = 11$ in Table 5 for

TABLE 5 Sparse grid results of a four- and five-dimensional put option on the maximum of the assets.

	European 4D			European 5D		
n_s	Price	Error	Ratio	Price	Error	Ratio
7	1.0488	5.25×10^{-2}		0.7407	3.48×10^{-2}	
8	1.0785	2.97×10^{-2}	1.77	0.7696	2.90×10^{-2}	1.20
9	1.0992	2.06×10^{-2}	1.44	0.7875	1.79×10^{-2}	1.62
10	1.1061	6.88×10^{-3}	3.00	0.7967	9.21×10^{-3}	1.94
11	1.1107	4.68×10^{-3}	1.47	0.8015	4.77×10^{-3}	1.93
12	1.1134	2.62×10^{-3}	1.79	0.8035	2.04×10^{-3}	2.34
13	1.1146	1.22×10^{-3}	2.15	0.8046	1.09×10^{-3}	1.34
	Bermudan 4D			Bermudan 5D		
n_s	Price	Error	Ratio	Price	Error	Ratio
10	1.830	1.34×10^{-2}	3.68	1.389	5.65×10^{-2}	0.32
11	1.838	5.09×10^{-3}	2.63	1.380	8.49×10^{-3}	6.66
12	1.840	3.18×10^{-3}	1.60	1.375	5.16×10^{-3}	1.65
13	1.841	2.56×10^{-3}	1.24	1.378	2.24×10^{-3}	2.30

the same option, but now in sparse grid case. In Table 6, the details are presented for the four-dimensional sparse grid case with $n_s = 11$. Each row in Table 6 represents a layer of the combination technique with the complexity, value of ℓ and number of subproblems. Columns four to six of Table 6 give the CPU times for each subproblem of a specific layer, the layer's total sequential CPU time and parallel CPU time when 12 CPUs are used. The total time on a single computer is 124.2 seconds and on a heterogeneous cluster 11.1 seconds. The efficiency is 11.23, which is high when 12 is the ideal case. We also see that the CPU time in the sparse grid context on 12 CPUs is lower than the CPU time for the full grid case on 16 CPUs (25 seconds, see Table 1). So, the sparse grid technique is an efficient method to use in parallel on a small number of CPUs, whereas the problem size of the partitioned full grid is still large. For example, the problem size of a subproblem in the top layer of the five-dimensional 2^{13} mimic in Table 5 has a total complexity of 2^{25} or 512 MB.

We conclude this section with sparse grid results with some higher-dimensional examples of the option contracts on the maximum or minimum of the assets. In Table 7, the results of the sparse grid computation are presented for a put option on the maximum and minimum of six or seven underlying assets. We again see a satisfactory accuracy with $n_s = 10$ but a highly irregular convergence. The seven-dimensional sparse grid problem with $n_s = 10$ uses the sparse grid technique as well as the partition technique in Section 3, because the maximum available memory is 2 GB on the heterogeneous cluster. The problem size of this experiment in the top layer is 4 GB and therefore it is partitioned with $B = 2$. Again the base of the sparse grid technique is set to $b = 3$. The hedge parameters can also be computed with the sparse grid technique. See Appendix B for the results for Δ and Γ.

TABLE 6 Details of the four-dimensional sparse grid computation with four layers (mimic of the four-dimensional 2^{11} full grid): the complexity of a single subproblem on layer ℓ, the number of subproblems, the CPU time for a subproblem and a layer (sequential and parallel with $Q = 12$).

Details			CPU times		
Layer ℓ	Complexity of a subproblem	Number of problems per layer	For a subproblem	Total for the layer	Parallel performance $Q = 12$
17	2^{20}	165	0.51	82.5	7.3
16	2^{19}	120	0.24	28.8	2.5
15	2^{18}	84	0.12	10.1	1.0
14	2^{17}	56	0.05	2.88	0.3
Total		425		124.2	11.1

TABLE 7 Sparse grid results of two types of six- and seven-dimensional European contracts.

	6D Put on minimum			6D Put on maximum		
n_S	Price	Error	Ratio	Price	Error	Ratio
7	27.093	1.43×10^{-1}		0.375	2.33×10^{-2}	
8	27.183	9.02×10^{-2}	1.58	0.396	2.13×10^{-2}	1.09
9	27.141	4.21×10^{-2}	2.14	0.412	1.50×10^{-2}	1.42
10	27.158	1.73×10^{-2}	2.43	0.420	8.89×10^{-3}	1.69
	7D Put on minimum			**7D Put on maximum**		
n_S	Price	Error	Ratio	Price	Error	Ratio
7	26.153	1.22×10^{-1}		0.179	1.45×10^{-2}	
8	26.217	6.31×10^{-2}	1.93	0.194	1.50×10^{-2}	0.96
9	26.189	2.72×10^{-2}	2.32	0.206	1.14×10^{-2}	1.31
10	26.203	1.34×10^{-2}	2.02	0.213	7.21×10^{-3}	1.58

5 CONCLUSIONS

The multi-dimensional convolution method is a powerful and fast method. It is able to efficiently price multi-asset options, especially those of European type, under Lévy price dynamics, including geometric Brownian motion, and to compute the hedge parameters. The inclusion of jumps in the underlying dynamics implies that we need to substitute another characteristic function in the multi-dimensional integration which is an easy exercise. However, the multi-dimensional models with jumps need insight into the correlation between the different assets, which is a non-trivial research question.

The partitioning of the method enables us to distribute some multi-dimensional split parts over a system of parallel computers, which speeds up the computation.

Since we chose to avoid communication in the parallel system, and thus require some additional computation, the parallel efficiency is not optimal.

With the help of the sparse grid technique, we can climb in the number of dimensions of the multi-dimensional contracts. The size of the target problem depends, of course, on the number of parallel processors that are at one's disposal. An important remark is, however, that the basic sparse grid technique, without any enhancements, can only be successfully applied for certain payouts, ie, solutions should have bounded mixed derivatives. We show that the min- and max-asset options exhibit a satisfactory sparse grid convergence. The parallel efficiency of the sparse grid method is excellent, as each subproblem can be computed independently. For high-dimensional problems it becomes necessary to combine the parallel sparse grid method with the parallel version of the convolution method.

For Bermudan options the convergence is irregular in the full grid case and rather slow in the sparse grid case. Bermudan options generally do not exhibit a smooth pasting between continuation value and early exercise payout. This hinders an efficient treatment, as at the early exercise point second derivatives do not exist and, in particular, mixed derivatives are not bounded. This is a prerequisite for an efficient sparse grid treatment. A logical next step in our research is to evaluate the resulting parallel method on a machine containing a significant number of parallel processors.

Although we have presented some positive results for the use of sparse grids for multi-asset options, the results also give rise to some serious thoughts on the applicability of the sparse grid method. Satisfactory sparse grid accuracy can be achieved for option values that exhibit bounded mixed derivatives. This may not, however, be possible for highly complex payout structures, as are usually encountered in the financial industry. For those, there is little hope for satisfactory sparse grid accuracy without any enhancements (making the method more complicated).

APPENDIX A TRUNCATION OF THE COMPUTATIONAL DOMAIN

Here we perform a numerical experiment in order to check numerically the influence of truncating the computational domain on the accuracy of option prices. We choose a single-asset option for this purpose. The option parameters used are $r = 0.06$, $\sigma = 0.25$, $\delta = 0.04$ and $T = 1$. The strike price is €40, as is $S(0)$, so $\log S_0/K = 0$. In Table A.1 we vary the size of the computational domain, Ω_d, and determine the error generated, where $S_{\min} = K \exp(-L)$ and $S_{\max} = K \exp(L)$.

We use 2^{20} discretization points, so that the discretization error is negligible. It is shown numerically that a domain of size $\Omega_d = [-L_1, L_1]$ with $L = 20\sigma$ gives highly accurate results.

APPENDIX B HEDGE PARAMETERS

The hedge parameters can be computed in an analytic way by using the derivative properties of the Fourier transform Gray and Goodman (1995):

$$\mathcal{F}\left(\frac{\mathrm{d}f}{\mathrm{d}x}\right) = -\mathrm{i}\omega\mathcal{F}(f)$$

TABLE A.1 At-the-money error caused by truncation to Ω_d with 2^{20} discretization points for a European call.

L	$S_{\min}$	$S_{\max}$	Error
σ	31.15	51.36	2.45×10^{0}
2σ	24.26	65.95	1.42×10^{0}
4σ	14.72	108.73	1.92×10^{-1}
8σ	5.41	295.56	2.40×10^{-4}
12σ	1.99	803.42	7.54×10^{-9}
16σ	0.73	2,183.91	6.86×10^{-10}
20σ	0.27	5,936.47	2.07×10^{-10}

Now the hedge parameters are:

$$\Delta_j(t, \mathbf{x}) = \frac{\partial V}{\partial S_j} = -\frac{\mathrm{e}^{-r(T-t)}}{S_j}\mathcal{F}^{\mathrm{inv}}\{\mathrm{i}\omega_j \mathcal{F}\{V(T, \mathbf{y})\}\phi(-\boldsymbol{\omega})\} \tag{B.1}$$

$$\Gamma_j(t, \mathbf{x}) = \frac{\partial^2 V}{\partial S_j^2} = \frac{\mathrm{e}^{-r(T-t)}}{S_j^2}\mathcal{F}^{\mathrm{inv}}\{(\mathrm{i}\omega_j + \omega_j^2)\mathcal{F}\{V(T, \mathbf{y})\}\phi(-\boldsymbol{\omega})\} \tag{B.2}$$

These equations can be discretized similarly to (26) into:

$$\Delta_k(t, \mathbf{x_m}) = -\frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d S_k}\prod_{j=1}^{d}(-1)^{m_j}\mathcal{D}_d^{\mathrm{inv}}[\mathrm{i}\omega_{n_k}\phi_{\mathbf{n}}\mathcal{D}_d\{V_{\mathbf{k}}G_{\mathbf{k}}\}] \tag{B.3}$$

$$\Gamma_k(t, \mathbf{x_m}) = \frac{\mathrm{e}^{-r(T-t)}}{(2\pi)^d S_k^2}\prod_{j=1}^{d}(-1)^{m_j}\mathcal{D}_d^{\mathrm{inv}}[(\mathrm{i}\omega_{n_k} + \omega_{n_k}^2)\phi_{\mathbf{n}}\mathcal{D}_d\{V_{\mathbf{k}}G_{\mathbf{k}}\}] \tag{B.4}$$

Table B.1 shows the numerical convergence results for the delta parameter in four- and five-dimensional basket call computations, obtained on a full grid. Table B.2 finally presents a comparison of the full and sparse grid numerical values for the parameter Δ.

TABLE B.1 Hedge parameters of a standard 4D and 5D basket call on a full grid of 2^{n_f} points per coordinate (option parameters are in Section 4.1)

	4D		5D	
n_f	Δ_1	Error	Δ_1	Error
3	0.1369		0.1087	
4	0.1370	8.90×10^{-4}	0.1119	3.29×10^{-3}
5	0.1371	1.16×10^{-4}	0.1116	3.40×10^{-4}
6	0.1372	2.22×10^{-5}	0.1116	1.90×10^{-5}

TABLE B.2 Hedge parameters for the four-dimensional put option on the minimum of the assets; full and sparse grid solutions.

	Full grid put on the minimum			
n_f	Δ_1	Error	Γ_1	Error
3	−0.2821	1.27×10^{-1}	1.126×10^{-2}	2.91×10^{-3}
4	−0.2322	4.99×10^{-2}	1.024×10^{-2}	1.03×10^{-3}
5	−0.2232	8.99×10^{-3}	9.823×10^{-3}	4.12×10^{-4}
6	−0.2223	9.91×10^{-4}	9.765×10^{-3}	5.80×10^{-5}
7	−0.2222	9.97×10^{-5}	9.751×10^{-3}	1.47×10^{-5}
	Sparse grid put on the minimum			
n_s	Δ_1	Error	Γ_1	Error
3	−0.2545		1.126×10^{-2}	
4	−0.2295	2.50×10^{-2}	9.262×10^{-3}	2.00×10^{-3}
5	−0.2209	8.63×10^{-3}	9.487×10^{-3}	2.25×10^{-4}
6	−0.2232	2.38×10^{-3}	9.661×10^{-3}	1.74×10^{-4}
7	−0.2211	2.18×10^{-3}	9.672×10^{-3}	1.14×10^{-4}
8	−0.2222	1.14×10^{-3}	9.774×10^{-3}	1.01×10^{-4}
9	−0.2224	2.14×10^{-4}	9.755×10^{-3}	1.86×10^{-5}
10	−0.2222	1.63×10^{-4}	9.752×10^{-3}	3.13×10^{-6}

REFERENCES

Bakshi, G., and Chen, Z. (1997). An alternative valuation model for contingent claims. *Journal of Financial Econometrics* **44**, 123–165.

Bellman, R. (1961). *Adaptive Control Processes; A Guided Tour*. Princeton University Press, Princeton, NJ.

Berridge, S. J., and Schumacher, J. W. (2004). An irregular grid method for high-dimensional free-boundary problems in finance. *Future Generation Computer Systems* **20**(3), 353–362.

bin Zubair, H., Leentvaar, C. C. W., and Oosterlee, C. W. (2007). Efficient d-multigrid preconditioners for sparse-grid solution of high-dimensional partial differential equations. *International Journal of Computational Mathematics* **8**, 1129–1147.

Bungartz, H. J., Griebel, M., Röschke, D., and Zenger, C. (1994). Pointwise convergence of the combination technique for the Laplace equation. *East–West Journal of Numerical Mathematics* **2**, 21–45.

Bungartz, H. J., and Griebel, M. (2004). Sparse grids. *Acta Numerica* May, 147–269.

Carr, P., and Madan, D. B. (1999). Option valuation using the fast Fourier transform. *The Journal of Computational Finance* **2**, 61–73.

Dempster, M. A. H., and Hong, S. S. G. (2000). Spread option valuation and the Fast Fourier Transform. Technical Report WP 26/2000, The Judge Institute of Management Studies, Cambridge University.

Garroni, M. G., and Menaldi, J. L. (1992). *Green Functions for Parabolic Second Order Integro-differential Equations* (*Pitman Research Notes in Mathematics Series, Vol. 275*). Longman, London.

Gentle, D. (1993) Basket weaving. *Risk* **6**, 51–52.

Gerstner, T., and Griebel, M. (1998). Numerical integration using sparse grids. *Numerical Algorithms* **18**, 209–232.

Gil-Pelaez, J. (1951). Note on the inverse theorem. *Biometrika* **37**, 481–482.

Gray, R. M., and Goodman, J. W. (1995). *Fourier Transforms*. Kluwer, Dordrecht.

Gurland, J. (1948). Inversion formulae for the distribution of ratios. *Annals of Mathematical Statistics* **19**, 228–237.

Leentvaar, C. C. W., and Oosterlee, C. W. (2008). On coordinate transformation and grid stretching for sparse grid pricing of basket options. *Journal of Computational and Applied Mathematics*, forthcoming.

Lord, R., Fang, F., Bervoets, F., and Oosterlee, C. W. (2008). A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes. *SIAM Journal on Scientific Computing* **30**, 1678–1705.

O'Sullivan, C. (2005). Path dependent option pricing under Lévy processes. EFA 2005 Moscow Meetings Paper, Available at SSRN: http://ssrn.com/abstract=673424.

Scott, L. (1997). Pricing stock options in a jump-diffusion model with stochastic volatility and interest rates: application of Fourier inversion methods. *Mathematical Finance* **7**, 413–426.

Singleton, K. J., and Umantsev, L. (2002). Pricing coupon-bond options and swaptions in affine term structure models. *Mathematical Finance* **12**(4), 427–446.

Tavella, D., and Randall, C. (2000). *Pricing Financial Instruments, the Finite Difference Method*. Wiley, New York.

Wu, J., Mehta, N. B., and Zhang, J. (2005). A flexible approximation of the sum of log-normal distributed values. *Global Telecommuications Conference (GLOBECOM)* **6**, 3413–3417.

Zenger, C. (1990). Sparse grids. In *Proceedings of the 6th GAMM Seminar* (*Notes on Numerical Fluid Mechanics, Vol. 31*). Friedrich Vieweg & Sohn, Braunschweig.

Zhu, Y., Wu, X., and Chern, I. (2004). *Derivative Securities and Difference Methods*. Springer, New York.