East Asian Journal on Applied Mathematics
doi: 10.4208/eajam.280313.061013a

Vol. **4**, No. 1, pp. 52-68
February 2014

Fast Exponential Time Integration for Pricing Options in Stochastic Volatility Jump Diffusion Models

Hong-Kui Pang[1] and Hai-Wei Sun[2,*]

[1] *School of Mathematics and Statistics, Jiangsu Normal University, Xuzhou, China.*
[2] *Department of Mathematics, University of Macau, Macao, China.*

Received 28 March 2013; Accepted (in revised version) 6 October 2013

Available online 24 February 2013

Abstract. The stochastic volatility jump diffusion model with jumps in both return and volatility leads to a two-dimensional partial integro-differential equation (PIDE). We exploit a fast exponential time integration scheme to solve this PIDE. After spatial discretization and temporal integration, the solution of the PIDE can be formulated as the action of an exponential of a block Toeplitz matrix on a vector. The shift-invert Arnoldi method is employed to approximate this product. To reduce the computational cost, matrix splitting is combined with the multigrid method to deal with the shift-invert matrix-vector product in each inner iteration. Numerical results show that our proposed scheme is more robust and efficient than the existing high accurate implicit-explicit Euler-based extrapolation scheme.

AMS subject classifications: 91B28, 62P05, 35K15, 65F10, 65M06, 91B70, 47B35

Key words: Stochastic volatility jump diffusion, European option, barrier option, partial integro-differential equation, matrix exponential, shift-invert Arnoldi, matrix splitting, multigrid method.

1. Introduction

In 1973, Black and Scholes [4] inytoduced a model to compute the price of a European option in the finance industry. Later empirical evidence indicated that the model assumptions on log-normality of the return of the underlying asset and constant volatility are usually inconsistent with market prices [24, 29]. Several extensions of the Black-Scholes model have been proposed. Examples include the jump diffusion [19, 24] or pure jump Lévy models [2, 6, 7, 11, 23], the stochastic volatility (SV) model [15, 18], the stochastic volatility with jumps in return (SVJ) model [3], and the stochastic volatility with correlated and contemporaneous jumps in return and variance (SVCJ) model [10, 13]. Of these, the SVCJ model usually offers a much better match with market prices than the others, since

*Corresponding author. *Email addresses:* panghongkui@163.com (H.-K. Pang), HSun@umac.mo (H.-W. Sun)

it allows the volatility to be stochastic and can capture the jumps in both the return and variance very well. In this article, we consider option valuation in the SVCJ model.

One way to price options involves solving a partial integro-differential equation (PIDE). For the SVCJ model, the corresponding PIDE is a two-dimensional equation involving a convolution integral defined over an infinite domain. D'Halluin *et al.* [9] proposed a second-order Crank-Nicolson scheme and Rannacher time-stepping to approximate PIDE, but direct application of the Crank-Nicolson scheme to the PIDE arising in the SVCJ model would suffer from the inversion of a block dense matrix at each time step. Andersen & Andreasen [1] used an operator-splitting approach combined with the fast Fourier transform (FFT) evaluation of a convolution integral to price European options with jump diffusion, but their method cannot easily handle the multi-dimensional jump diffusion processes and processes with state-dependent jump magnitude distributions — cf. [14]. Recently, an extrapolation scheme based on the implicit-explicit (IMEX) Euler method was proposed by Feng & Linetsky [14], to deal with the jump-diffusion PIDE, where the differential term is treated implicitly for stability and the integral term explicitly for numerical efficiency. Combined with the extrapolation approach, this approach is remarkably fast and can achieve polynomial accuracy in the time direction. Zhang *et al.* [34] further improved the efficiency of the extrapolation scheme by coupling quadratic finite elements for spatial discretization with preconditioning techniques for the resulting systems. It is known that the IMEX Euler-based extrapolation scheme belongs to the class of time-stepping methods. When the time interval is large, many time steps may be required in order to achieve a given accuracy, which is very time consuming. Instead of using a time-stepping scheme, some other authors have proposed to exploit the exponential time integration (ETI) scheme to solve PIDEs arising from both the Black-Scholes and Merton models [28, 32]. The ETI scheme is a one-step method, and we need not consider stability nor temporal discretized accuracy.

This article discusses the application of the ETI scheme for pricing options in the SVCJ model, such that the price of an option involves the product of a matrix exponential and a vector. In Refs. [28,32], the corresponding matrix exponential was directly computed by a scaling and squaring algorithm with Padé's approximation [16], which has $O(n^3)$ complexity where n is the matrix size, whereas we need to find the product of a matrix exponential and a vector but not the exact matrix exponential where the recently developed Krylov subspace methods often work very well [12, 17, 22, 25, 33]. Quite recently, Lee, Liu & Sun [20] employed the shift-invert Arnoldi method proposed in Ref. [21], to implement the ETI scheme for option pricing in the Merton [24] and Kou jump-diffusio [19] models. By exploiting the Toeplitz structure of the resulting matrix and the Gohberg-Semencul formula (GSF) for inversion of the Toeplitz matrix, the computational cost of the ETI scheme was reduced dramatically to $O(n \log n)$ operations. It is notable that the PIDE arising in the SVCJ model has two space variables related to the asset price and the volatility, respectively. After the spatial discretization of the PIDE by central differences and the time integration of the semi-discretized ODE system by the ETI scheme, we obtain a solution vector as the product of the exponential of a block Toeplitz matrix and a vector. When the shift-invert Arnoldi method is applied to approximate this resulting solution vector, a

shifted and inverted block Toeplitz matrix multiplied by a vector must be dealt with in each iteration step. In passing, we note that the resulting matrix is block Toeplitz but not exact Toeplitz, so there is no efficient inversion formula like the GSF for the standard Toeplitz matrix. Thus an inner-outer iteration is needed, and a matrix splitting technique combined with a multigrid method is proposed for the inner iteration In order to reduce the computational burden.

The rest of this paper is arranged as follows. In Section 2, we recall the PIDE formulation for options valuation in the SVCJ model and perform both the spatial discretization and time integration. In Section 3, we introduce the shift-invert Arnoldi method and present an algorithm for fast implementation of the ETI scheme. Numerical comparisons between the ETI scheme and the IMEX Euler-based extrapolation scheme are discussed in Section 4, and our concluding remarks are in Section 5.

2. Discretization of the PIDE in the SVCJ Model

Let us first recall the PIDE arising in the SVCJ model — cf. [13,14,34] for more details. Thus if $u(t,x,y)$ denotes the option price of a European-style option at time $\bar{T}-t$, the underlying asset price is Ke^x and its instantaneous variance is $(1+y)\theta$, where $\bar{T}$ refers to the maturity time, K the strike price, and θ the long-run variance level. respectively. The option value function $u(t,x,y)$ in the SVCJ model then satisfies the following two-dimensional PIDE [14,34]:

$$u_t = \mathscr{A}u + \mathscr{B}u\,, \qquad t\in(0,\bar{T}]\,, \qquad (x,y)\in\Omega\,, \tag{2.1}$$

where

$$\begin{aligned}\mathscr{A}u &= \frac{1}{2}\theta\left(y+1\right)u_{xx} + \rho_D\zeta\left(y+1\right)u_{xy} + \frac{\zeta^2}{2\theta}\left(y+1\right)u_{yy}\\ &\quad + \left[\mu - \frac{1}{2}\theta\left(y+1\right)\right]u_x - \kappa y u_y - (r+\lambda)u\,,\\ \mathscr{B}u &= \lambda\int_{-\infty}^{\infty}\int_{0}^{\infty} u\left(t,x+z^x,y+z^y\right)p(z^x,z^y)\mathrm{d}z^y\,\mathrm{d}z^x\,,\end{aligned}$$

with the joint probability density

$$p(z^x,z^y) = \frac{\theta}{\nu\sqrt{2\pi\sigma_J^2}}\exp\left[-\frac{\theta z^y}{\nu} - \frac{\left(z^x-\mu_J-\rho_J\theta z^y\right)^2}{2\sigma_J^2}\right], \quad z^x\in\mathbb{R},\ z^y\geq 0\,. \tag{2.2}$$

Here the parameter ρ_D is the coefficient correlating Brownian shocks in the return and variance processes, ζ is the volatility-of-volatility parameter,

$$\mu = r - q + \lambda\left[1-(1-\nu\rho_J)^{-1}\exp(\mu_J+\sigma_J^2/2)\right]$$

is the drift parameter, κ is the rate of mean reversion, r is the risk-free interest rate, q is the dividend rate, λ is the intensity of compound Poisson process, and ν, μ_J, ρ_J, σ_J are parameters related to jumps in the return and variance.

As stated in Refs. [14,34], the spatial domain Ω in (2.1) depends on the type of option, and in this article we consider the European vanilla option and the double barrier option. Theoretically, the domain is $\Omega = (-\infty, +\infty) \times (-1, +\infty)$ for the vanilla option and $\Omega = (x_L, x_R) \times (-1, +\infty)$ for the double barrier option, with lower barrier L and upper barrier U where $x_L = \ln(L/K)$ and $x_R = \ln(U/K)$. To render the numerical solution feasible, the spatial domain is restricted to a bounded set $(x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$ with $y_{\min} = -1$ and fixed $x_{\min}$, $x_{\max}$, and $y_{\max}$ chosen sufficiently large. Specifically, $x_{\min} = x_L$ and $x_{\max} = x_R$ for the double barrier option, where for convenience the bounded set $(x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$ is still denoted by Ω.

The initial condition for (2.1) is given by

$$u(0, x, y) = \psi(x, y)\,, \qquad (x, y) \in \Omega$$

with $\psi(x, y)$ being the payoff function. For a European call option $\psi(x, y) = K(e^x - 1)^+$, whereas for a put option $\psi(x, y) = K(1 - e^x)^+$, where $x^+ = \max\{x, 0\}$. Dirichlet boundary conditions are imposed on the complement of the bounded set $\Omega = (x_{\min}, x_{\max}) \times (y_{\min}, y_{\max})$ [14,34]:

$$u(t, x, y) = \phi(x, y), \qquad (x, y) \in \Omega^c,$$

where Ω^c represents the complement of Ω in the whole plane. In general, $\phi = \psi$ for the European vanilla option and $\phi = 0$ for the double barrier option — cf. [14, 34] for more details.

In the following, we show how to discretise the PIDE (2.1) and transform it into a matrix exponential problem.

2.1. Spatial discretization

Let the spatial domain of (2.1) be restricted to $\Omega = (x_{\min},\ x_{\max}) \times (y_{\min},\ y_{\max})$ and define a computational grid $\Omega_{h_x, h_y} = \{(x_i, y_j), 0 \le i \le m, 0 \le j \le n\}$ with

$$\begin{cases} x_i = x_{\min} + i h_x\,, \quad h_x = \dfrac{x_{\max} - x_{\min}}{m}\,, \quad i = 0, 1, \cdots, m\,, \\ y_j = y_{\min} + j h_y\,, \quad h_y = \dfrac{y_{\max} - y_{\min}}{n}\,, \quad j = 0, 1, \cdots, n\,. \end{cases}$$

We approximate the differential term $\mathscr{A}u$ in (2.1) by second-order central differences. Let $u_{i,j}(t)$ denote the approximation to $u(t, x_i, y_j)$, and $\mathbf{u}(t) = [\mathbf{u}_1(t), \cdots, \mathbf{u}_{m-1}(t)]^\intercal$ with $\mathbf{u}_i(t) = [u_{i,1}(t), \cdots, u_{i,n-1}(t)]$ for $i = 1, \cdots, m-1$. Then the differential term $\mathscr{A}u$ has the discretization form

$$A\mathbf{u}(t) + \mathbf{f}_A\,,$$

where A is a block tri-diagonal Toeplitz matrix

$$A = \begin{bmatrix} A_{(0)} & A_{(1)} & & 0 \\ A_{(-1)} & A_{(0)} & \ddots & \\ & \ddots & \ddots & A_{(1)} \\ 0 & & A_{(-1)} & A_{(0)} \end{bmatrix} \in \mathbb{R}^{(m-1)(n-1)\times(m-1)(n-1)} \tag{2.3}$$

with tri-diagonal blocks $A_{(-1)}, A_{(0)}, A_{(1)} \in \mathbb{R}^{(n-1)\times(n-1)}$ given respectively by

$$\begin{aligned}
A_{(-1)} &= D_{n-1} \cdot \text{tridiag}\left(\frac{\rho_D\zeta}{4h_xh_y}, \frac{\theta}{2h_x^2} + \frac{\theta}{4h_x}, -\frac{\rho_D\zeta}{4h_xh_y}\right) - \frac{\mu}{2h_x}I\,,\\
A_{(0)} &= D_{n-1} \cdot \text{tridiag}\left(\frac{\zeta^2}{2\theta h_y^2} + \frac{\kappa}{2h_y}, -\frac{\theta}{h_x^2} - \frac{\zeta^2}{\theta h_y^2}, \frac{\zeta^2}{2\theta h_y^2} - \frac{\kappa}{2h_y}\right)\\
&\quad + \text{tridiag}\left(-\frac{\kappa}{2h_y}, -(r+\lambda), \frac{\kappa}{2h_y}\right),\\
A_{(1)} &= D_{n-1} \cdot \text{tridiag}\left(-\frac{\rho_D\zeta}{4h_xh_y}, \frac{\theta}{2h_x^2} - \frac{\theta}{4h_x}, \frac{\rho_D\zeta}{4h_xh_y}\right) + \frac{\mu}{2h_x}I\,,
\end{aligned}$$

$D_{n-1} = \text{diag}(y_1 + 1, \cdots, y_{n-1} + 1)$, I is the identity matrix, and $\mathbf{f}_A$ is the load vector. For the barrier option, we note that $\mathbf{f}_A$ is zero due to the zero boundary condition.

For the integral term $\mathscr{B}u$, at each inner grid point (x_i, y_j), $i = 1, \cdots, m-1$ and $j = 1, \cdots, n-1$, we first change variables and then split the integral into two parts:

$$\begin{aligned}
\mathscr{B}u(t, x_i, y_j) &= \lambda \int_{-\infty}^{\infty}\int_{0}^{\infty} u\left(t, x_i + z^x, y_j + z^y\right) p(z^x, z^y)\mathrm{d}z^y\,\mathrm{d}z^x\\
&= \lambda \int_{-\infty}^{\infty}\int_{y_j}^{\infty} u\left(t, \omega, \eta\right) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega\\
&= \lambda \int_{x_{\min}}^{x_{\max}}\int_{y_j}^{y_{\max}} u\left(t, \omega, \eta\right) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega\\
&\quad + \lambda \iint_{\Omega_j^*} u\left(t, \omega, \eta\right) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega\,,
\end{aligned} \tag{2.4}$$

where $\Omega_j^* = (-\infty, \infty) \times (y_j, \infty) \backslash (x_{\min}, x_{\max}) \times (y_j, y_{\max})$. For the first part of (2.4), we use the composite trapezoidal rule with step sizes h_x and h_y in the x and y directions respectively, and obtain the approximation

$$\begin{aligned}
&\lambda \int_{x_{\min}}^{x_{\max}}\int_{y_j}^{y_{\max}} u\left(t, \omega, \eta\right) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega\\
\approx &\lambda\frac{h_xh_y}{4}\Bigg(p_{-i,0}u_{0,j} + p_{m-i,0}u_{m,j} + p_{-i,n-j}u_{0,n} + p_{m-i,n-j}u_{m,n} + 2\sum_{k=1}^{m-1} p_{k-i,0}u_{k,j}\\
&+ 2\sum_{k=1}^{m-1} p_{k-i,n-j}u_{k,n} + 2\sum_{l=j+1}^{n-1} p_{-i,l-j}u_{0,l} + 2\sum_{l=j+1}^{n-1} p_{m-i,l-j}u_{m,l} + 4\sum_{k=1}^{m-1}\sum_{l=j+1}^{n-1} p_{k-i,l-j}u_{k,l}\Bigg)\,,
\end{aligned}$$

where $p_{k-i,l-j} = p(x_k - x_i, y_l - y_j)$ for $k = 0, 1, \cdots, m$ and $l = 0, 1, \cdots, n$. Rewriting all these approximations in matrix form yields the vector

$$B\mathbf{u}(t) + \mathbf{f}_{B_1}\,,$$

where the matrix B is a block Toeplitz matrix with Toeplitz blocks (BTTB) given by

$$B = \begin{bmatrix} B_{(0)} & B_{(1)} & \cdots & B_{(m-2)} \\ B_{(-1)} & B_{(0)} & \ddots & \vdots \\ \vdots & \ddots & \ddots & B_{(1)} \\ B_{(-(m-2))} & \cdots & B_{(-1)} & B_{(0)} \end{bmatrix} \in \mathbb{R}^{(m-1)(n-1)\times(m-1)(n-1)}, \tag{2.5}$$

in which each block $B_{(k)}$ has the form

$$B_{(k)} = \begin{bmatrix} b_0^k & b_1^k & \cdots & b_{n-2}^k \\ 0 & b_0^k & \ddots & \vdots \\ \vdots & \ddots & \ddots & b_1^k \\ 0 & \cdots & 0 & b_0^k \end{bmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}$$

with $b_0^k = \frac{1}{2}\lambda h_x h_y p_{k,0}$ and $b_l^k = \lambda h_x h_y p_{k,l}$, $k = 0, \pm 1, \cdots, \pm(m-2)$, $l = 1, \cdots, n-2$, and $\mathbf{f}_{B_1}$ is the load vector. Again, for the double barrier option the vector $\mathbf{f}_{B_1}$ is zero given the zero boundary condition.

The value of the second part in (2.4) completely depends on the option contract to be priced. If it is the double barrier option, then the value is naturally equal to zero because of the zero boundary condition $\phi = 0$; but If it is the vanilla put option, then the option price $u(t,x,y)$ over Ω_* is given by $\phi(x,y) = K(1-e^x)^+$, and thus

$$\begin{aligned} &\lambda \iint_{\Omega_j^*} u\left(t,\omega,\eta\right) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega \\ =&\lambda \int_{-\infty}^{x_{\min}} \int_{y_j}^{\infty} K(1-e^{\omega}) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega \\ &+ \lambda \int_{x_{\min}}^{0} \int_{y_{\max}}^{\infty} K(1-e^{\omega}) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega\,. \end{aligned} \tag{2.6}$$

On substituting the probability density function (2.2) into (2.6), after some calculations we obtain

$$\begin{aligned} &\lambda \int_{-\infty}^{x_{\min}} \int_{y_j}^{\infty} K(1-e^{\omega}) p(\omega - x_i, \eta - y_j)\mathrm{d}\eta\mathrm{d}\omega \\ =&\frac{\lambda K\theta}{\nu}\left(\int_0^{\infty} e^{-\frac{\theta\xi}{\nu}} \mathcal{N}(\chi_i(\xi))\mathrm{d}\xi - e^{x_i + \frac{\sigma_J^2}{2} + \mu_J} \int_0^{\infty} e^{(\rho_J - \frac{1}{\nu})\theta\xi} \mathcal{N}(\chi_i(\xi) - \sigma_J)\mathrm{d}\xi\right) \end{aligned}$$

and

$$\begin{aligned}
&\lambda\int_{x_{\min}}^{0}\int_{y_{\max}}^{\infty}K(1-e^{\omega})p(\omega-x_i,\eta-y_j)\mathrm{d}\eta\mathrm{d}\omega\\
=&\frac{\lambda K\theta}{\nu}\left(\int_{y_{\max}-y_j}^{\infty}e^{-\frac{\theta\xi}{\nu}}\big(\mathcal{N}\big(\chi_i(\xi)-\frac{x_{\min}}{\sigma_J}\big)-\mathcal{N}(\chi_i(\xi))\big)\mathrm{d}\xi\right.\\
&\left.-e^{x_i+\frac{\sigma_J^2}{2}+\mu_J}\int_{y_{\max}-y_j}^{\infty}e^{(\rho_J-\frac{1}{\nu})\theta\xi}\big(\mathcal{N}\big(\chi_i(\xi)-\frac{x_{\min}+\sigma_J^2}{\sigma_J}\big)-\mathcal{N}(\chi_i(\xi)-\sigma_J)\big)\mathrm{d}\xi\right),
\end{aligned}$$

where $\mathcal{N}(\cdot)$ is the standard normal cumulative distribution function and

$$\chi_i(\xi)=\frac{x_{\min}-x_i-\mu_J-\rho_J\theta\xi}{\sigma_J},\qquad i=1,2,\cdots,m-1\,.$$

Since the integrands in above integrals decay rapidly as the variable of integration ξ approaches infinity, we can evaluate these integrals efficiently using a numerical integration method such as the adaptive Gauss-Kronrod quadrature method [31]. The value of (2.6) obtained is taken as the $((i-1)(n-1)+j)$-th element of a vector $\mathbf{f}_{B_2}$. For the vanilla call option, the second part in (2.4) can also be computed similarly.

After collecting all these discretization terms together, we obtain the semi-discretized form of (2.1) as the ODE system

$$\frac{\mathrm{d}\mathbf{u}(t)}{\mathrm{d}t}=A\mathbf{u}(t)+B\mathbf{u}(t)+\mathbf{f}\,,\qquad t\in(0,\bar{T}\,],\tag{2.7}$$

where A and B are respectively given by (2.3) and (2.5), and $\mathbf{f}=\mathbf{f}_A+\mathbf{f}_{B_1}+\mathbf{f}_{B_2}$.

2.2. Time integration

Rather than employ a time-stepping method for (2.7), as previously mentioned we consider the ETI scheme used in Refs. [20, 28, 32] for option pricing problems in one-dimensional models. Thus on integrating (2.7) over the time interval $(0,\bar{T}\,]$, we obtain the solution

$$\mathbf{u}(\bar{T})=e^{\bar{T}(A+B)}\mathbf{u}(0)+\int_0^{\bar{T}}e^{(\bar{T}-t)(A+B)}\mathbf{f}\mathrm{d}t$$

at the maturity time $\bar{T}$, where $\mathbf{u}(0)$ is the initial vector containing the discrete values of the payoff function $\psi(x,y)$. From the assumptions on the boundary conditions, we see that the vector $\mathbf{f}$ is independent of the time t, so the solution can be explicitly written as

$$\mathbf{u}(\bar{T})=e^{\bar{T}(A+B)}\left(\mathbf{u}(0)+(A+B)^{-1}\mathbf{f}\right)-(A+B)^{-1}\mathbf{f}\,.\tag{2.8}$$

Unlike a time-stepping method, the ETI scheme is exact and does not require discretization in the time direction.

With reference to (2.8), in order to evaluate the option price $\mathbf{u}(\bar{T})$ at the maturity time $\bar{T}$ one has to compute both the term $(A+B)^{-1}\mathbf{f}$ and the product of the matrix exponential $e^{\bar{T}(A+B)}$ and a vector. From (2.3) and (2.5) we know that both matrices A and B in (2.8) are block Toeplitz matrices, and therefore so is $A+B$. Thus the term $(A+B)^{-1}\mathbf{f}$ can be computed using preconditioned Krylov subspace methods with block circulant preconditioners [8], or other fast solvers such as the multigrid method [5]. Moreover, this term vanishes ($\mathbf{f}=0$) for the double barrier option. Consequently, the main workload in (2.8) is to compute the product of a block Toeplitz matrix exponential and a vector. As mentioned in Section 1, direct computation of the matrix exponential is prohibitive given its cubic operation cost with respect to the matrix size [16], so to improve the performance of the ETI scheme we adopt the shift-invert Arnoldi method coupled with a multigrid solver to efficiently approximate the product $e^{\bar{T}(A+B)}v$ for some vector v.

3. Shift-Invert Arnoldi Method

3.1. Summary of the shift-invert Arnoldi method

Over the past two decades, the class of Krylov subspace methods to deal efficiently with the product of a large matrix exponential and a vector have been investigated intensively [12, 17, 22, 25, 33]. The standard approach to approximate the product $e^{T_n}v$ with $T_n \in \mathbb{R}^{n\times n}$ and $v \in \mathbb{R}^n$ is to first establish an orthonormal basis $V_m = [v_1, v_2, \cdots, v_m]$ of the Krylov subspace $\mathscr{K}_m(T_n, v) = \text{span}\{v, T_n v, \cdots, T_n^{m-1} v\}$, which can be done using the Arnoldi process for a non-symmetric matrix or the Lanczos process for a symmetric matrix. The result is summarised in the matrix formulation

$$T_n V_m = V_m H_m + h_{m+1,m} v_{m+1} e_m^* ,$$

where H_m is the m-by-m projected matrix, e_m is the m-th canonical basis vector and v_{m+1} is a unit vector that satisfies $V_m^* v_{m+1} = 0$. Thus the vector $e^{T_n}v$ is approximated by [30]

$$e^{T_n} v \approx \beta V_m e^{H_m} e_1 , \qquad \beta = \|v\|_2 ,$$

where $\|\cdot\|_2$ is the spectral norm. It is clear that the Krylov subspace method reduces the problem of approximating the vector $e^{T_n}v$ where T_n is a large n-by-n matrix to that of computing the vector $e^{H_m}e_1$ where H_m is a smaller m-by-m matrix. Moreover, only the matrix-vector product is needed during the whole process.

When the Krylov subspace method is implemented, one naturally hopes that the iteration number m is as small as possible, in order to achieve an acceptable accuracy. However, theoretical analyses [17, 30] show that the error decay is generally related to the norm $\|T_n\|_2$, so the number of iterations m can become quite large as $\|T_n\|_2$ increases. To accelerate the approximation process of the Krylov subspace method, in Ref. [25, 33] the authors proposed to use the shift-invert preconditioning technique. The motivation for this comes from the observation that the exponential function is quickly decaying and the vector $e^{T_n}v$ is mostly determined by small eigenvalues and their corresponding invariant

subspaces, provided the spectrum of T_n is contained in the left half-plane. Implementation of this preconditioning technique is as follows. Firstly, the product $e^{T_n}v$ is rewritten as

$$e^{T_n}v = e^{-\frac{1}{\gamma}(((I-\gamma T_n)^{-1})^{-1}-I)}v\,,$$

where I is the identity matrix and $\gamma > 0$ is a shift parameter. We then construct an orthonormal basis $\widehat{V}_m = [\hat{v}_1, \hat{v}_2, \cdots, \hat{v}_m]$ of the Krylov subspace $\mathscr{K}_m((I-\gamma T_n)^{-1}, v)$ using the Arnoldi (or Lanczos) process, which leads to the following relations:

$$(I-\gamma T_n)^{-1}\widehat{V}_m = \widehat{V}_m\widehat{H}_m + \hat{h}_{m+1,m}\hat{v}_{m+1}e_m^*, \qquad \widehat{V}_m^*\hat{v}_{m+1} = 0\,.$$

Finally, the vector $e^{-\frac{1}{\gamma}(((I-\gamma T_n)^{-1})^{-1}-I)}v$ is approximated by

$$e^{-\frac{1}{\gamma}(((I-\gamma T_n)^{-1})^{-1}-I)}v \approx \beta\widehat{V}_m e^{-\frac{1}{\gamma}(\widehat{H}_m^{-1}-I)}e_1\,, \qquad \beta = \|v\|_2\,. \tag{3.1}$$

The whole process is summarised in the following algorithm:

Algorithm 3.1: Shift-invert Arnoldi method for $e^{T_n}v$.

1. Initialize: Compute $\beta = \|v\|_2$ and $\hat{v}_1 = v/\beta$
2. Iterate: Do $j = 1, \cdots, m$
   (a) Compute $w := (I-\gamma T_n)^{-1}\hat{v}_j$
   (b) Do $k = 1, \cdots, j$
      i. Compute $\hat{h}_{k,j} := (w, \hat{v}_k)$
      ii. Compute $w := w - \hat{h}_{k,j}\hat{v}_k$
   (c) Compute $\hat{h}_{j+1,j} := \|w\|_2$ and $\hat{v}_{j+1} := w/\hat{h}_{j+1,j}$
3. Approximation: $e^{T_n}v \approx \beta\widehat{V}_m e^{-\frac{1}{\gamma}(\widehat{H}_m^{-1}-I)}e_1$

Both theoretical analyses and numerical experiments in show that this shift-invert preconditioning technique can speed up the approximation process dramatically, provided that the matrix T_n is a sectorial operator [25, 33]. However, it is not easy to prove theoretically that the semi-discrete matrix $A+B$ is a sectorial operator. In Fig. 1, we plot the eigenvalues of $A+B$ with the grid numbers $m = 16$ and $n = 128$ for the European vanilla put option, where model parameters are taken from Table 1. We can see from Fig. 1 that all the eigenvalues are located in the left half plane. Numerical experiments in Section 4 also show the efficiency of the shift-invert Arnoldi method for our problems.

According to Algorithm 3.1, we have to compute the product $(I-\gamma T_n)^{-1}\hat{v}_j$ in each iteration step. If T_n is a Toeplitz matrix, then the computation of this product via the celebrated Toeplitz inverse formula GSF [20, 21, 26] by FFT is very fast. However, in our case the matrix T_n is the sum of block Toeplitz matrices A and B but not an exact Toeplitz matrix, so the GSF cannot be used and we have to solve the linear system

$$\big(I-\gamma(A+B)\big)w = \hat{v}_j \tag{3.2}$$

in each iteration step. In the following subsection, we employ matrix splitting combined with a multigrid method to solve this system efficiently.

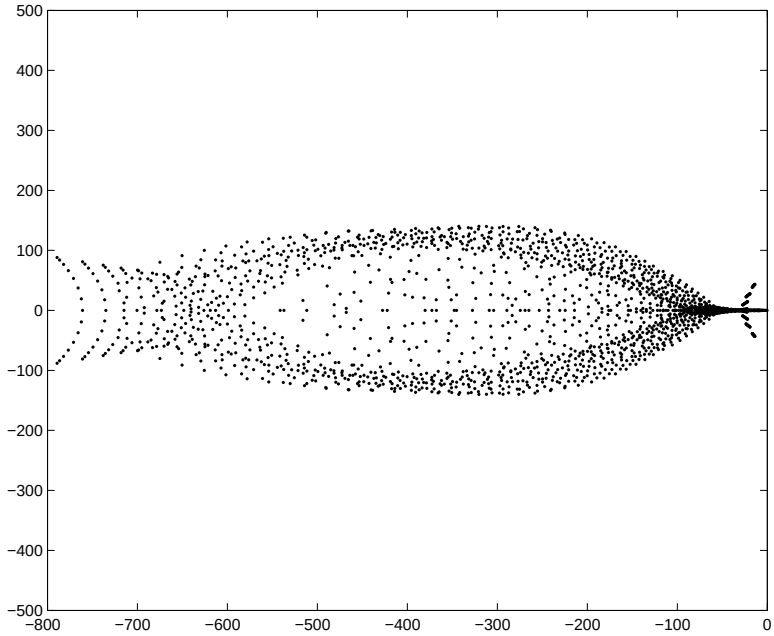

Figure 1: Eigenvalues of matrix $A+B$ with $m=16$ and $n=128$.

3.2. Matrix splitting technique combined with a multigrid method

We recall that the matrix A in (3.2) is a block tri-diagonal Toeplitz matrix with tri-diagonal blocks, and B is a BTTB dense matrix. To avoid the inversion of a large dense matrix, we put B in the right-hand side of (3.2) and construct the iteration scheme

$$(I-\gamma A)w_{k+1}=\gamma Bw_k+\hat{v}_j\,,\qquad k=0,1,\cdots\,, \tag{3.3}$$

where w_0 is an initial guess. Next, we exploit a multigrid method to solve these systems.

Multigrid methods are a class of very efficient algorithms for solving linear systems arising from PDE, where the basic idea is to employ relaxations on a fine grid to damp oscillatory errors and approximations on a coarser grid to smooth errors. For a general linear system

$$A_N w=b \tag{3.4}$$

with $A_N\in\mathbb{C}^{N\times N}$ and $b\in\mathbb{C}^N$, a commonly used procedure in the multigrid method is the V-cycle. Thus suppose we have provided a sequence of coarse grid operators A_{N_s} with $1\le s\le l$, the relaxation operators $G_{N_s}\colon \mathbb{C}^{N_s}\to\mathbb{C}^{N_s}$ with ν_1 and ν_2 being the numbers of pre- and post-relaxation steps respectively, the restriction operators $I_s^{s+1}:\mathbb{C}^{N_s}\to\mathbb{C}^{N_{s+1}}$, the interpolation operators $I_{s+1}^{s}\colon \mathbb{C}^{N_{s+1}}\to\mathbb{C}^{N_s}$, as well as the coarsest grid number N_l. Here l is the total number of grid levels, with $s=1$ being the finest level — i.e. for $s=1$, $A_{N_1}=A_N$. Then the V-cycle multigrid algorithm can be described as below [5]:

Algorithm 3.2: V-cycle.

Procedure MG(ν_1, ν_2) (w_s, b_s)
If $s = l$
$w_l := A_{N_l}^{-1} b_l$;
else
for $j = 1 : \nu_1$
$w_s := G_{N_s} w_s + (I - G_{N_s}) A_{N_s}^{-1} b_s$;
end
$d_{s+1} := I_s^{s+1}(b_s - A_{N_s} w_s)$;
$e_{s+1}^0 := 0$;
e_{s+1} :=MG(ν_1, ν_2) (e_{s+1}^0, d_{s+1});
$w_s := w_s + I_{s+1}^s e_{s+1}$;
for $j = 1 : \nu_2$
$w_s := G_{N_s} w_s + (I - G_{N_s}) A_{N_s}^{-1} b_s$;
end
end
MG(ν_1, ν_2):= w_s;
end

When Algorithm 3.2 is applied to solve the k-th system of (3.3), then the finest grid operator, the unknown vector and the right-hand side in (3.4) become

$$A_N = I - \gamma A\,, \quad w = w_{k+1} \quad \text{and} \quad b = \gamma B w_k + \hat{v}_j\,,$$

where $N = (m-1)(n-1)$. We performed the V-cycle with full coarsening and set the total grid level $l = \log_2(\min\{m,n\})$, with the coarse grid number $N_s = (\frac{m}{2^{s-1}} - 1)(\frac{n}{2^{s-1}} - 1)$ for $1 \le s \le l$. We noted that the matrix A is a block tri-diagonal Toeplitz matrix with tri-diagonal blocks, and so is A_N, which inspired us to use the block Gauss-Seidel relaxation operator [5]. In addition, we adopted the red-black strategy, which does not add any computational cost. To retain the block tri-diagonal Toeplitz structure in coarse grid operators, we constructed A_{N_s} by discretising the PIDE (2.1) with coarse grid numbers. In other words, we defined the coarse grid operators in geometric means.

The restriction operators were defined as

$$I_s^{s+1} = P \otimes P$$

with

$$P = \frac{1}{4}\begin{bmatrix} 1 & 2 & 1 & & & & & \\ & & 1 & 2 & 1 & & & \\ & & & & & \ddots & & \\ & & & & & & 1 & 2 & 1 \end{bmatrix},$$

and interpolation operators as $I_{s+1}^s = 4 I_s^{s+1}$, $1 \le s \le l$ (cf. [5]). When solving the k-th system of (3.3), we took the approximation solution of the $(k-1)$-th system as the initial

guess. Moreover, only one V-cycle was carried out to approximate the solution of each system in (3.3), and our numerical results in the next section show that one V-cycle is enough for the convergence of the iteration scheme (3.3).

Finally, we considered the computational cost of solving the iteration scheme (3.3). To solve the k-th system of (3.3), we to first worked out the right-hand side $b = \gamma B w_k + \hat{v}_j$. As matrix B is a BTTB matrix, the vector b can be constructed by the FFT with

$$\mathrm{O}\big(mn\log(mn)\big)$$

operations [8]. When the V-cycle in Algorithm 3.2 is applied to the the k-th system of (3.3), the coefficient matrix A_{N_s} on each grid level is a block tri-diagonal matrix with tri-diagonal blocks and the relaxation adopted in the V-cycle is the block Gauss-Seidel iteration, such that the system on each grid level can be solved in

$$\mathrm{O}(N_s) = \mathrm{O}\left(\frac{mn}{4^{s-1}}\right)$$

operations. Thus the computational cost per V-cycle multigrid iteration for solving the k-th system of (3.3) is roughly

$$\left(1+\frac{1}{4}+\frac{1}{4^2}+\cdots+\frac{1}{4^{l-1}}\right)\cdot \mathrm{O}\big((m-1)(n-1)\big) = \mathrm{O}(mn)$$

provided that the right hand side has been obtained. On the whole, the total computational cost for solving the k-th system of (3.3) is about

$$\mathrm{O}\big(mn\log(mn)\big)$$

operations. Furthermore, if the iteration scheme (3.3) converges linearly, then the total computational cost for solving (3.2) is $\mathrm{O}(mn\log(mn))$ operations, which implies that the computational cost in each iteration step of the shift-invert Arnoldi method is also $\mathrm{O}(mn\log(mn))$ operations. The linear convergence rate of the iteration scheme (3.3) has not been proved in this paper. However, numerical results in the following section show that the scheme (3.3) converges very fast and the convergence rate is almost independent of grid numbers.

4. Numerical Experiments

In this section, we demonstrate numerically the efficiency of the proposed ETI scheme by comparing with the IMEX Euler-based extrapolation scheme introduced in Ref. [14]. All experiments are performed in MATLAB 7.11 (R2010b) on a PC with the configuration Intel(R)Xeon(R)CPU W3520 @ 2.67GHz and 6.00GB RAM.

We considered the European vanilla put (EVP) option and the knock-out double barrier put (DBP) option under the SVCJ model. The corresponding payoff function is $\psi(x,y) = K(1-e^x)^+$. Parameter values in our numerical experiments were taken from Refs. [14,34]

and given in Table 1. Among them, λ, ν, μ_J, σ_J, ρ_D, ρ_J, ζ, κ, θ, r, q are model parameters and K is the strike price, and L and U denote the lower and upper barriers of the DBP option, respectively. In practice, the infinite domains of EVP and DBP options are restricted to the bounded domains Ω_{EVP} and Ω_{DBP}, respectively. We call these bounded domains the computational domain. Numerical tests show that the computational domains given in Table 1 lead to small enough truncation errors for the considered option prices.

Table 1: Parameter values used in numerical experiments.

Model and option parameters	$\lambda = 4,\ \nu = 0.02,\ \mu_J = -0.04,\ \sigma_J = 0.06,\ \rho_D = -0.5,\ \rho_J = -0.5$ $\zeta = 0.1,\ \kappa = 4,\ \theta = 0.04,\ r = 5\%,\ q = 2\%,\ K = 100,\ L = 80,\ U = 120$
Other parameters	$\Omega_{DBP} = (x_{\min}, x_{\max}) \times (y_{\min}, y_{\max}) = (\ln 0.8, \ln 1.2) \times (-1, 7)$, $\Omega_{EVP} = (x_{\min}, x_{\max}) \times (y_{\min}, y_{\max}) = (-0.8, 0.8) \times (-1, 7)$.

As foreshadowed, the ETI scheme was implemented by the the shift-invert Arnoldi method. In all tests, the shift parameter chosen was $\gamma = 1/15$, the stopping criterion via the formula (3.19) in Refs. [27], and the related tolerance was 10^{-5}. In the inner iteration of the shift-invert Arnoldi process, we employed the matrix splitting iteration with multigrid method as described in Subsection 3.2, to solve the system (3.2). Only one V-cycle was carried out at each step of the iteration scheme (3.3), and the number of pre-correction and post-correction relaxation sweeps in the V-cycle were both set to be 1. For comparison, we also used the same V-cycle to solve the systems arising from the IMEX Euler-based extrapolation scheme [14]. The extrapolation process was stopped when the local error tolerance became less than 10^{-5}. In both cases, we stopped solving the resulting systems when the relative residual error was less that 10^{-8}.

We first priced a 3 month ($\bar{T} = 0.25$) EVP option, which can also be done by inverting the Fourier transform by FFT [10]. We investigated the pricing error at the at-the-money option with the asset price $S_0 = K = 100$ and the volatility $\sigma_0 = 20\%$ — which corresponds to the point $(x, y) = (0, 0)$, since the asset price S, the volatility σ, x, and y have the relationships $S = Ke^x$, $\sigma = \sqrt{\theta\,(y+1)}$. The benchmark price was computed using the FFT and is 4.812582536. Table 2 reports the numerical results of the ETI scheme and the extrapolation scheme with different grid numbers. In the table, the symbol "(m,n)" represents the number of spatial grid points, "ETI" and "Extrapolation" respectively refer to the ETI scheme and the extrapolation scheme, and "outer/inner" denotes the out iteration number and the average inner iteration number. For the ETI scheme, the outer iteration number refers to iteration steps of the Arnoldi process, and the inner iteration number refers to the average iteration number of the iteration scheme (3.3). For the extrapolation scheme, the outer iteration number refers to the number of resulting systems required to be solved in the extrapolation process, and the inner iteration number refers to the average iteration number of solving those systems. "CPUtime" denotes the total CPU time of performing the corresponding schemes with the unit in second. The label "Error" in the table represents the pricing error at the at-the-many option (i.e., at the point $(x, y) = (0, 0)$), and the column "ratio" is the ratio of two consecutive pricing errors.

Numerical results in columns "Error" and "ratio" show that the numerical solutions

Table 2: Numerical results for pricing the EVP option with maturity time $\bar{T} = 0.25$ by the ETI scheme and the extrapolation scheme respectively.

(m,n)	ETI		Extrapolation		Error	ratio
	out/inner	CPUtime	out/inner	CPUtime		
(16, 128)	11/6.09	0.15	45/5.22	0.37	4.20e-001	4.45
(32, 256)	14/6.00	0.46	45/7.11	1.38	9.44e-002	4.13
(64, 512)	14/6.07	1.65	45/7.89	4.49	2.29e-002	4.01
(128,1024)	16/7.00	8.47	45/8.00	16.31	5.70e-003	4.03
(256,2048)	18/7.06	41.22	45/8.00	68.75	1.41e-003	

converge to the benchmark and the convergence rate is the second-order accurate, which exemplifies the correction of our proposed scheme. From Table 2, in order to reach the same accuracy we see that the outer iteration number and the CPU time required by the ETI scheme are much less than that by the extrapolation scheme. In addition, the average inner iteration numbers for the ETI scheme are not large and are almost independent of spatial grid numbers, which implies that the splitting iteration scheme (3.3) converges very fast, and in each iteration step one V-cycle is enough to approximate the solution.

To further illustrate the efficiency of the ETI scheme on other numerical aspects, we also priced 3 month ($\bar{T} = 0.25$), 6 month ($\bar{T} = 0.5$), and 1 year ($\bar{T} = 1.0$) DBP options. The pricing errors were investigated in the approximation domain $G = (\underline{x}_G, \bar{x}_G) \times (\underline{y}_G, \bar{y}_G) = (\ln 0.8, \ln 1.2) \times (0, 3)$ (corresponding to $(80, 120)$ in the underlying asset price and $(20\%, 40\%)$ in the volatility), where we are interested in the value function u. We computed the benchmark price with large enough grid numbers ($m = 320$ and $n = 2048$) using the ETI scheme. Tables 3, 4 and 5 display the numerical results for pricing DBP options with maturity times $\bar{T} = 0.25$, $\bar{T} = 0.5$ and $\bar{T} = 1.0$ respectively. In these tables, "Error" refers to the maximum norm pricing error, which was computed over all grid points of the approximation domain G. Other symbols have the same meaning as in Table 2. Numerical results in Tables 3, 4, and 5 demonstrate again that the proposed ETI scheme outperforms the IMEX Euler-based extrapolation scheme in both outer iteration number and CPU time, especially when the maturity time $\bar{T}$ is large. Moreover, with increasing of $\bar{T}$ the outer iteration numbers required by the ETI scheme do not increase, while those from the extrapolation scheme do, another advantage of the ETI scheme over the extrapolation scheme.

5. Concluding Remarks

In this article, we considered approximating the product of a matrix exponential with block Toeplitz matrix and a vector, arising from integrating a two-dimensional PIDE in option pricing problems in the SVCJ model by the ETI scheme. The shift-invert Arnoldi method was employed to compute this product efficiently, which results in an inner-outer iteration. To relieve the computational burden, the matrix splitting technique with multi-

Table 3: Numerical results for pricing the DBP option with maturity time $\bar{T} = 0.25$ by the ETI scheme and the extrapolation scheme respectively.

(m,n)	ETI		Extrapolation		Error	ratio
	out/inner	CPUtime	out/inner	CPUtime		
(10, 64)	9/6.00	0.07	55/7.45	0.37	4.26e-002	2.60
(20, 128)	18/8.00	0.37	55/9.22	0.87	1.64e-002	4.13
(40, 256)	18/9.50	1.13	55/9.85	2.25	3.96e-003	4.35
(80, 512)	18/10.67	4.52	55/10.51	7.68	9.11e-004	5.27
(160,1024)	18/11.11	18.86	55/10.73	30.81	1.73e-004	
(320,2048)	19/11.63	77.73	55/10.95	132.22		

Table 4: Numerical results for pricing the DBP option with maturity time $\bar{T} = 0.5$ by the ETI scheme and the extrapolation scheme respectively.

(m,n)	ETI		Extrapolation		Error	ratio
	out/inner	CPUtime	out/inner	CPUtime		
(10, 64)	15/6.93	0.13	66/8.18	0.51	2.12e-002	4.63
(20, 128)	18/8.89	0.42	66/10.02	1.13	4.58e-003	4.20
(40, 256)	16/10.00	1.10	66/10.55	3.31	1.09e-003	4.52
(80, 512)	16/10.88	4.30	66/10.94	10.42	2.41e-004	5.62
(160,1024)	16/11.50	17.64	66/11.27	39.43	4.29e-005	
(320,2048)	16/11.88	66.78	66/11.45	166.29		

Table 5: Numerical results for pricing the DBP option with maturity time $\bar{T} = 1$ by the ETI scheme and the extrapolation scheme respectively.

(m,n)	ETI		Extrapolation		Error	ratio
	out/inner	CPUtime	out/inner	CPUtime		
(10, 64)	11/8.45	0.11	78/8.79	0.61	1.43e-002	14.57
(20, 128)	14/9.64	0.34	78/10.63	1.44	9.84e-004	3.84
(40, 256)	14/10.43	0.97	91/10.92	4.17	2.56e-004	3.84
(80, 512)	11/11.18	2.96	91/11.40	13.60	6.66e-005	4.46
(160,1024)	14/11.93	15.82	91/11.52	53.07	1.49e-005	
(320,2048)	14/12.14	59.84	91/11.71	229.14		

grid method was proposed to deal with the shift-invert block Toeplitz matrix-vector product in each inner iteration. Our numerical results to demonstrate the efficiency of the overall proposed scheme. There remain some problems that should be considered in future. One is to prove theoretically that the semi=discrete matrix $A+B$ is a sectorial operator, to guarantee the fast convergence rate of the shift-invert Arnoldi method, and another is to analyse rigorously the convergence of the splitting iteration scheme (3.3).

Acknowledgments

The author is grateful to the anonymous reviewers for their comments and their effort in improving the paper. First author was supported by the National Natural Science Foundation of China under grant 11201192, the Natural Science Foundation of Jiangsu Province under grant BK2012577, and the Natural Science Foundation for Colleges and Universities in Jiangsu Province under grant 12KJB110004. Second author was supported by research grants 033/2009/A and 005/2012/A1 from FDCT and MYRG206(Y1-L4)-FST11-SHW from University of Macau.

References

[1] L. Andersen and J. Andreasen, *Jump-diffusion processes: Volatility smile fitting and numerical methods for option pricing*, Rev. Derivatives Res. **4**, 231–262 (2000).
[2] O. Barndorff-Nielsen, *Process of normal inverse Gaussian type*, Finance Stoch. **2**, 41–68 (1998).
[3] D. Bates, *Jumps and stochastic volatility: Exchange rate processes implicit in Deutsche Mark options*, Rev. Financial Stud. **9**, 69–107 (1996).
[4] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, J. Polit. Econ. **81**, 637–654 (1973).
[5] W.L. Briggs, V.E. Henson, and S.F. McCormick, *A Multigrid Tutorial*, 2nd edition, SIAM, Philadelphia, 2000.
[6] P. Carr, H. Geman, D. Madan, and M. Yor, *The fine structure of asset returns: An empirical investigation*, J. Business **75**, 305–332 (2002).
[7] P. Carr and L. Wu, *Finite moment log stable process and option pricing*, J. Finance **58**, 753–777 (2003).
[8] R. Chan and X. Jin, *An Introduction to Iterative Toeplitz Solvers*, SIAM, Philadelphia, 2007.
[9] Y. d'Halluin, P. Forsyth, and K. Vetzal, *Robust numerical methods for contingent claims under jump diffusion*, IMA J. Numer. Anal. **25**, 87–112 (2005).
[10] D. Duffie, J. Pan, and K. Singleton, *Transform analysis and asset pricing for affine jump diffusions*, Econometrica **68**, 1343–1376 (2000).
[11] E. Eberlein, U. Keller, and K. Prause, *New insights into smile, mispricing, and value at risk: The hyperbolic model*, J. Business **71**, 371–405 (1998).
[12] M. Eiermann and O. Ernst, *A restarted Krylov subspace method for the evaluation of matrix functions*, SIAM J. Numer. Anal. **44**, 2481–2504 (2006).
[13] B. Eraker, M. Johannes, and N. Polson, *The impact of jumps in volatility and returns*, J. Finance **58**, 1269–1300 (2003).
[14] L. Feng and V. Linetsky, *Pricing options in jump diffusion models: An extrapolation approach*, Oper. Res. **56**, 304–325 (2008).
[15] S. Heston, *A closed form solution for options with stochastic volatility with applications to bond and currency options*, Rev. Financial Stud. **6**, 327–343 (1993).
[16] N. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.
[17] M. Hochbruck and C. Lubich, *On Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal. **34**, 1911–1925 (1997).
[18] J. Hull and A. White, *The pricing of options on assets with stochastic volatilites*, J. Finance **42**, 281–300 (1987).
[19] S. Kou, *A jump diffusion model for option pricing*, Management Sci. **48**, 1086–1101 (2002).

[20] S. Lee, X. Liu, and H. Sun, *Fast exponential time integration scheme for option pricing with jumps*, Numer. Linear Algebra Appl. **19**, 87–101 (2012).
[21] S. Lee, H. Pang and H. Sun, *Shift-invert Arnoldi approximation to the Toeplitz matrix exponential*, SIAM J. Sci. Comput. **32**, 774–792 (2010).
[22] L. Lopez and V. Simoncini, *Analysis of projection methods for rational function approximation to the matrix exponential*, SIAM J. Numer. Anal. **44**, 613–635 (2006).
[23] D. Madan, P. Carr, and E. Chang, *The variance gamma process and option pricing*, Eur. Finance Rev. **2** 79–105 (1998).
[24] R. Merton, *Option pricing when underlying stock returns are discontinuous*, J. Financial Econom. **3**, 125–144 (1976).
[25] I. Moret and P. Novati, *RD-rational approximations of the matrix exponential*, BIT, 44 (2004), pp. 595–615.
[26] H. Pang and H. Sun, *Shift-invert Lanczos method for the symmetric positive semidefinite Toeplitz matrix exponential*, Numer. Linear Algebra Appl., 18 (2011), pp. 603–614.
[27] M. Popolizio and V. Simoncini, *Acceleration techniques for approximating the matrix exponential operator*, SIAM J. Matrix Anal. Appl., 30 (2008), pp. 657–683.
[28] N. Rambeerich, D. Tangman, A. Gopaul, and M. Bhuruth, *Exponential time integration for fast finite element solutions of some financial engineering problems*, J. Comput. Appl. Math. **224**, 668–678 (2009).
[29] M. Rubinstein, *Implied binomial trees*, J. Finance **49**, 771–818 (1994).
[30] Y. Saad, *Analysis of some Krylov subspace approximations to the matrix exponential operator*, SIAM J. Numer. Anal. **29**, 209–228 (1992).
[31] L. Shampine, *Vectorized adaptive quadrature in Matlab*, J. Comput. Appl. Math. **211**, 131–140 (2008).
[32] D. Tangman, A. Gopaul, and M. Bhuruth, *Exponential time integration and Chebychev discretisation schemes for fast pricing of options*, Appl. Numer. Math. **58**, 1309–1319 (2008).
[33] J. van den Eshof and M. Hochbruck, *Preconditioning Lanczos approximations to the matrix exponential*, SIAM J. Sci. Comput. **27**, 1438–1457 (2006).
[34] Y. Zhang, H. Pang, L. Feng, and X. Jin, *Quadratic finite element and preconditioning for options pricing in the SVCJ model*, to appear in J. Comput. Finance.