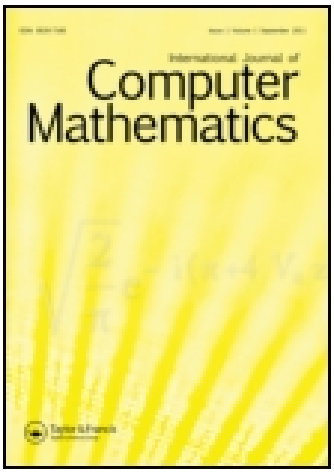

International Journal of Computer Mathematics
Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/gcom20

Comparison and survey of finite difference methods for pricing American options under finite activity jump-diffusion models

Santtu Salmi [a] & Jari Toivanen [a b]
[a] Department of Mathematical Information Technology , University of Jyväskylä , PO Box 35, Agora , FI 40014 , Finland
[b] Institute for Computational and Mathematical Engineering, Stanford University , Stanford , CA , 94305 , USA
Published online: 22 Mar 2012.

To cite this article: Santtu Salmi & Jari Toivanen (2012) Comparison and survey of finite difference methods for pricing American options under finite activity jump-diffusion models, International Journal of Computer Mathematics, 89:9, 1112-1134, DOI: 10.1080/00207160.2012.669475

To link to this article: http://dx.doi.org/10.1080/00207160.2012.669475

International Journal of Computer Mathematics
Vol. 89, No. 9, June 2012, 1112–1134

Comparison and survey of finite difference methods for pricing American options under finite activity jump-diffusion models

Santtu Salmi[a]* and Jari Toivanen[a,b]

[a]*Department of Mathematical Information Technology, University of Jyväskylä, PO Box 35 (Agora), FI 40014, Finland;* [b]*Institute for Computational and Mathematical Engineering, Stanford University, Stanford, CA 94305, USA*

(*Received 12 September 2011; revised version received 13 February 2012; accepted 20 February 2012*)

Partial integro-differential formulations are often used for pricing American options under jump-diffusion models. A survey on such formulations and their numerical methods is presented. A detailed description of six efficient methods based on a linear complementarity formulation and finite difference discretizations is given. Numerical experiments compare the performance of these methods for pricing American put options under finite activity jump models.

Keywords: jump-diffusion model; American option; linear complementarity problem; finite difference method; iterative method

2010 AMS Subject Classifications: 35K85; 65M06; 65Y20; 91G20; 91G60

1. Introduction

Since the seminal papers by Black and Scholes [7] and by Merton [42] in 1973, it has become evident that their underlying model is not consistent with the market prices of options. In particular, the so-called implied volatility typically exhibits a smile-like shape with respect to the strike price, which becomes more pronounced as the maturity approaches.

The addition of jumps into the model of the underlying asset leads to such a volatility smile [3,13]. First, such a model has already been introduced in 1976 by Merton [43]. Other jump models include the CGMY model [12] and Kou's model [35]. The models can be divided into two categories: finite and infinite jump activity models. In this paper, we consider models with finite jump activity, which include Merton's and Kou's jump-diffusion models. In [4,17], models with jumps and stochastic volatility were considered. Here, we restrict our study to models with a deterministic volatility, which can be a constant or a function of time, and the value of the underlying asset [3].

Based on a jump-diffusion model, a partial integro-differential equation (PIDE) can be derived for the price of a European option. Similarly, the price of an American option is given by a partial

*Corresponding author. Email: santtu.salmi@jyu.fi

ISSN 0020-7160 print/ISSN 1029-0265 online

http://dx.doi.org/10.1080/00207160.2012.669475
http://www.tandfonline.com

integro-differential inequality, which can be formulated as a linear complementarity problem (LCP). In this paper, we consider American options. While in a special case, the price of an American option can be presented as a series [58], the price is usually computed by discretizing the continuous problem and solving the resulting algebraic problems. Most commonly, finite differences are used for discretizing the partial derivatives. We also adopt this approach here. An alternative approach is to employ the characteristic function of the jump operator together with an efficient fast Fourier transformation (FFT)-based implementation; see [10,18,19,38], for example.

The discretization of the integral term in a jump-diffusion model leads to a full matrix. Due to this, special discretizations and algorithms have to be developed in order to obtain the option prices fast. Under finite activity jump models, the jump integral is a compact operator. Employing this property, efficient discretizations and solution algorithms can be derived. This paper surveys the research on this topic. The integral term can be treated implicitly or explicitly in the time discretization. Furthermore, LCPs can be solved directly or iteratively, transformed into a penalty form, or approximated using operator splitting. Combinations of these lead to six different numerical methods, which are described and compared in this paper. To our knowledge, two of these methods have not been presented previously in the scientific literature: the explicit treatment of the integral term with a direct solver for the resulting LCPs and the explicit treatment of the integral term with a penalty approximation for the resulting LCPs.

2. Survey

This section surveys research related to formulations and numerical methods based on partial integro-differential operators for pricing American options under finite activity jump-diffusion models. For more general models, finite difference methods have been discussed for American options in [53].

2.1 *Formulations*

The most common way is to formulate a partial integro-differential inequality which has a form of an LCP for the price of an American option. Zhang [57] described a variational inequality for the price under the Merton model. It can be shown to be equivalent to the LCP form [33]. Alternatively, these problems can be formulated with a Lagrange multiplier as has been done in [31,32].

The LCP can be approximated using a penalty method. The penalty is based on the violation of the early exercise constraint which requires the price to be at least the same as the payoff of the option. Under the Black–Scholes model, these methods have been considered in [21,44], and under the jump-diffusion models, they have been used in [23,52]. This approach leads to a nonlinear problem without any constraints.

Under finite activity jump models, the so-called smooth pasting principle holds [46], which states that the price of an American option is smooth across the early exercise boundary. Based on this principle, a free boundary problem can be stated for the price. Employing this formulation, the so-called front-tracking and front-fixing methods can be formulated. Under the Black–Scholes model, early studies on these methods include [44,45,56] and the more recent ones include [25,60]. Under jump-diffusion models, this approach has been considered in [54].

The price of an American option can also be obtained as the solution of a linear programming problem [16]. To our knowledge, this approach has not been used under jump-diffusion models. Yet another approach is to derive a semilinear equation for the price [6]. For jump-diffusion models, this formulation has been presented in [34].

2.2 *Discretizations*

The most common way to discretize the spatial derivatives is to use finite differences. For American options under the Black–Scholes model, they had been used already in 1977 by Brennan and Schwartz [8]. Under the Merton model, they were used in [57]. Typically, second-order accurate central finite differences are used. More recently, finite element methods have been applied for pricing options; see [1,25], for example. In particular, it has been more common to perform error analysis for finite element discretizations. Also, spectral element methods have been employed during the recent years like in [59]. The integral term resulting from the jumps in the model is often approximated using a suitable quadrature rule, for example, the trapezoidal rule. In some cases, the integral is calculated analytically under the assumption that the price function is polynomial between grid points.

The Crank–Nicolson method is the most common time discretization method. As the payoff function is usually non-smooth, the lack of L-stability of the Crank–Nicolson method can cause oscillations in the price function. One way to avoid this is to employ the Rannacher time scheme [22,48], which takes a few first time steps with the implicit Euler method and then uses the Crank–Nicolson method. Also, many other time discretizations have been employed in the option pricing context. Usually, explicit methods are not used as the small time steps required by their stability leads to poor computational efficiency. Some examples of other implicit methods are the backward differentiation formulas and the Runge–Kutta methods. More recently, exponential time integrations have also been studied in [47].

With an implicit time discretization, the integral term due to the jumps leads to algebraic problems with full matrices. It can be challenging to solve such problems efficiently. Special time discretizations circumventing this have been proposed. They treat the integral term explicitly and the rest of the operator implicitly. Under finite activity jump models, the compactness of the jump operator makes stability restrictions fairly relaxed, thus allowing efficient time step sizes. Such first-order accurate schemes were used in [14,57]. Second-order accurate implicit-explicit (IMEX) methods were employed in [9,59]. A better suited second-order method for pricing American options has recently been proposed by Kwon and Lee [36,37]. A similar second-order IMEX midpoint scheme was used for European options by Feng and Linetsky [20].

Operator splitting methods decompose the spatial operator into two or more parts and treat them differently in the time stepping. For example, in [3], an alternating direction implicit (ADI) type method was proposed to treat the integral term on its own fractional time step. An operator splitting method treating the early exercise constraint for American options was proposed under the Black–Scholes model in [28]. Since then, several papers including [28,30,37,47,52] have used this approach for pricing American options under various models.

2.3 *Solution of the resulting algebraic systems*

With the linear complementarity formulation, implicit time discretizations require solving an LCP at each time step. A popular iterative method for these LCPs is the projected successive over relaxation (PSOR) method [15]; see also [26]. With a jump model, the resulting matrices are full and, thus, PSOR iterations are computationally very expensive. A policy iteration for computing the solution of the time-dependent LCP was proposed in [5]. In this method, a sequence of time-dependent LCPs over the whole time interval under a model without jumps are solved. In [50], a similar iteration was derived algebraically for each time step. In this method, at each time step, the solution of an LCP with a full matrix is obtained by solving a sequence of LCPs with a banded matrix. A more generic form of this iteration was considered in [27]. Typically, the banded matrix is obtained from a model without jumps. These methods require multiplying vectors by the full matrix resulting from

the jump part. A different type of method based on a fixed-point iteration was proposed in [40].

The iterations proposed in [5,50] require solutions of LCPs with banded, typically tridiagonal, matrices. While the PSOR method can be used for these problems, the Brennan–Schwartz algorithm [8] offers a more efficient alternative when applicable; see also [1,29,33]. This direct method can be obtained by adding a projection in the back-substitution steps when solving a system of linear equations using an $\boldsymbol{LU}$ (or $\boldsymbol{UL}$) decomposition. The projection is performed on each component immediately after it is computed using the $\boldsymbol{U}$ matrix by replacing it by the payoff value if it is less than the payoff. The Brennan–Schwartz algorithm has optimal computational complexity and, in practice, it is very fast.

The penalty approximation for an LCP leads to a system of nonlinear equations. A traditional way to solve such nonlinear systems is to use a semismooth Newton method. It requires solving linear problems with the Jacobian matrix, which is a full matrix under jump models. While in this case, such linear problems can be solved efficiently, it is even more efficient to employ an approximate semismooth Newton method proposed in [23]; see also [52]. It approximates the Jacobian matrix by a matrix resulting from a model without jumps. Thus, it requires only solutions of linear problems with a banded matrix and multiplications by the full matrix resulting from the jump term.

2.4 *Computation of the integral term*

All previously mentioned solution methods require the computation of the integral term, which in the discrete case leads to a multiplication of a vector by a full matrix. As the matrix is full, the straightforward implementation of the multiplication is computationally expensive. More precisely, the number of operations is proportional to the number of grid points squared. Thus, on finer grids required for good accuracy, this computational cost can be prohibitively high.

Usually, the integral term can be transformed into a convolution form. When the grid for the convolution integral is uniform, which typically means a uniform log price grid, the convolutions can be computed exactly and efficiently using two FFTs. The computational cost of an FFT is the number of grid points times the logarithm of it, which is quite close to the optimal cost and very fast in practice. When grids are not uniform, additional interpolations are introduced between uniform and non-uniform grids in order to use the FFT. The accuracy and efficiency of this interpolation approach depend on how fine the uniform grid is. One needs to make a compromise between speed and accuracy. The use of the FFT in the computation of the integrals is the most popular approach and it has been used in [2,3,23,24,49–51,54], for example.

A wavelet compression technique was considered for computing the integrals in [39–41]. This is an approximation which leads to optimal or near-optimal computational costs. Uniform grids have been used with this technique and in the case of non-uniform grids and interpolations have been employed in a manner that is the same as that used with the FFT. For Kou's model, which has a log-double-exponential jump distribution, it is possible to construct recursion formulas to compute the integrals with optimal computational costs also on non-uniform grids. This approach has been studied in [11,52].

3. Option pricing model

In this section, we describe the mathematical model for pricing European and American options under a jump-diffusion process. We assume that the log return of the stock price S_t follows a

finite activity jump-diffusion process as in [13]. Then, the stochastic differential equation for S_t is given by

$$\frac{\mathrm{d}S_t}{S_{t-}} = (r - \lambda\kappa)\,\mathrm{d}t + \sigma\,\mathrm{d}W(t) + (\eta - 1)\,\mathrm{d}N_t, \tag{1}$$

where r is the risk-free interest rate, σ is the volatility, $W(t)$ is a standard Brownian motion, N_t is a Poisson process with rate λ, $(\eta - 1)$ is a random variable of the jump distribution producing a jump from S_{t-} to ηS_{t-}, and κ is the expected value of $(\eta - 1)$. The notation S_{t-} means that whenever there is a jump, the value of the process before the jump is used.

Two well-known finite activity jump-diffusion models in the literature are the log-normal model proposed by Merton [43] and the log-double-exponential model proposed by Kou [35]. Under Merton's model, jump sizes follow a log-normal distribution with the density function

$$f_{\ln}(y) := \frac{1}{y\delta\sqrt{2\pi}} \exp\left(-\frac{(\log y - \gamma)^2}{2\delta^2}\right). \tag{2}$$

Under Kou's model, jump sizes follow a log-double-exponential distribution with the density function

$$f_{\mathrm{ld}}(y) := \begin{cases} q\alpha_2 y^{\alpha_2 - 1}, & y < 1, \\ p\alpha_1 y^{-\alpha_1 - 1}, & y \geq 1, \end{cases} \tag{3}$$

where $p, q, \alpha_1 > 1$, and α_2 are positive constants such that $p + q = 1$.

The expected value of $(\eta - 1)$ under the probability distribution f is given by

$$E[\eta - 1] = \int_0^\infty (\eta - 1) f(\eta)\,\mathrm{d}\eta. \tag{4}$$

For the expected relative jump sizes, we have $\kappa = E[\eta - 1] = \exp(\gamma + \delta^2/2) - 1$ for Merton's model, and $\kappa = E[\eta - 1] = p\alpha_1/(\alpha_1 - 1) + q\alpha_2/(\alpha_2 + 1) - 1$ for Kou's model.

Now, let $v(x, \tau)$ be the value of a European option contract that depends on the price of the underlying asset $x = S_t$ and the time to maturity $\tau = T - t$. The function $v(x, \tau)$ satisfies the PIDE

$$v_\tau = \frac{1}{2}\sigma^2 x^2 v_{xx} + (r - \lambda\kappa) x v_x - (r + \lambda) v + \lambda \left(\int_0^\infty v(xy, \tau) f(y)\,\mathrm{d}y\right) =: Lv \tag{5}$$

for all $(x, \tau) \in [0, \infty) \times (0, T]$, where f is the density function of the jump distribution [43,55]. The value of v at maturity is given by

$$v(x, T) = g(x), \quad x \in [0, \infty), \tag{6}$$

where $g(x)$ is the payoff function for the option contract. For a put option, it is

$$g(x) = \max(K - x, 0), \tag{7}$$

where K is the strike price. The boundary conditions for a European put option are given by

$$\begin{aligned} v(0, \tau) &= K\,\mathrm{e}^{-r\tau}, \\ \lim_{x \to \infty} v(x, \tau) &= 0, \quad \tau \in [0, T]. \end{aligned} \tag{8}$$

The value v of an American put option can be obtained by solving the LCP

$$\begin{aligned} (v_\tau - Lv) &\geq 0, \quad v \geq g, \\ (v_\tau - Lv)(v - g) &= 0 \end{aligned} \tag{9}$$

with the boundary conditions

$$\begin{aligned} v(0,\tau) &= K, \\ \lim_{x\to\infty} v(x,\tau) &= 0, \quad \tau \in [0,T]. \end{aligned} \tag{10}$$

See [26] for more information.

4. Discretization for European and American options

In the following, we consider a finite difference discretization of Equation (5). For ease of notation, we divide the spatial operator of the PIDE (5) into three parts as follows:

$$Lv = Dv + Jv - (r+\lambda)v, \tag{11}$$

where D is the differential operator and J is the integral operator defined by

$$Dv := \tfrac{1}{2}\sigma^2 x^2 v_{xx} + (r - \lambda\kappa)xv_x, \tag{12}$$

$$Jv := \lambda\left(\int_0^\infty v(xy,\tau)f(y)\,\mathrm{d}y\right). \tag{13}$$

The infinite space domain $[0,\infty)$ is truncated into $[0,X]$ with a sufficiently large X. A uniform grid

$$0 = x_1 < x_2 < \cdots < x_N = X \tag{14}$$

with N nodes is used.

The spatial derivatives of the differential operator D are approximated with central differences

$$v_x = v_x(x_n,\tau) \approx \frac{v(x_{n+1},\tau) - v(x_{n-1},\tau)}{2\Delta x} \tag{15}$$

and

$$v_{xx} = v_{xx}(x_n,\tau) \approx \frac{v(x_{n+1},\tau) - 2v(x_n,\tau) + v(x_{n-1},\tau)}{(\Delta x)^2}, \tag{16}$$

where $\Delta x = X/(N-1)$.

The main challenge lies in the efficient treatment of the integral operator J. Traditionally, second-order accurate time discretizations are implicit schemes which lead to solving problems with full matrices due to the jump operator J. An explicit treatment of J leads to solving problems with tridiagonal matrices and multiplications by full matrices.

We transform the integral J into a convolution integral; see [2,24], for example. We can then compute the matrix–vector multiplication by employing an FFT which requires $O(N\log N)$ operations. For Kou's model, we employ recursion formulas described in [11,52]. This technique has the optimal computational complexity of $O(N)$.

We price American options by numerical methods that employ two or three time levels in the time discretization. In the following, we describe these discretizations.

4.1 *Time discretization with two time levels*

We use the Rannacher scheme [48] employing the implicit Euler method and the Crank–Nicolson method. For the PIDE (5), this scheme reads

$$\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_m}{\Delta\tau} = \boldsymbol{D}(\theta\boldsymbol{v}_{m+1}+(1-\theta)\boldsymbol{v}_m) + \boldsymbol{J}(\theta\boldsymbol{v}_{m+1}+(1-\theta)\boldsymbol{v}_m) - (r+\lambda)(\theta\boldsymbol{v}_{m+1}+(1-\theta)\boldsymbol{v}_m), \tag{17}$$

where $\boldsymbol{v}_m$ is the vector $\boldsymbol{v}_m = (v(x_1,m), v(x_2,m), \ldots, v(x_N,m))^{\mathrm{T}}$, $\Delta\tau$ is the time step size $\Delta\tau = \Delta\tau_m = \tau_{m+1}-\tau_m$, $\boldsymbol{D}$ is the discrete differential operator, $\boldsymbol{J}$ is the discrete integral operator, and θ is an indicator function defined by

$$\theta = \theta_m = \begin{cases} 1 & \text{for an implicit Euler time step,} \\ \frac{1}{2} & \text{for a Crank–Nicolson time step.} \end{cases} \tag{18}$$

For American options, we obtain the discrete LCP

$$\begin{aligned} &\left(\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_m}{\Delta\tau} - \boldsymbol{L}_2\boldsymbol{v}_m\right) \geq \boldsymbol{0}, \quad \boldsymbol{v}_m \geq \boldsymbol{g}, \\ &\left(\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_m}{\Delta\tau} - \boldsymbol{L}_2\boldsymbol{v}_m\right)^{\mathrm{T}} (\boldsymbol{v}_m - \boldsymbol{g}) = 0, \end{aligned} \tag{19}$$

where $\boldsymbol{g}$ is the discrete payoff function and $\boldsymbol{L}_2\boldsymbol{v}_m$ is the right-hand side of Equation (17). For a more detailed explanation of the discretization, see [50,52].

4.2 *Time discretization with three time levels*

We consider the three-level implicit–explicit time discretization described by Kwon and Lee [36,37]. After the first time step, central differences are used for time over three time levels. The differential operator is approximated by the average of the first and third time levels, and the integral operator is applied at the second time level. Thus, the differential part is treated implicitly, while the integral part is treated explicitly. This scheme for the PIDE (5) reads

$$\begin{aligned} \frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_m}{\Delta\tau} &= \boldsymbol{D}\boldsymbol{v}_{m+1} + \boldsymbol{J}\boldsymbol{v}_m - (r+\lambda)\boldsymbol{v}_{m+1} \quad \text{for } m=1, \\ \frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_{m-1}}{2\Delta\tau} &= \boldsymbol{D}\left(\frac{\boldsymbol{v}_{m+1}+\boldsymbol{v}_{m-1}}{2}\right) + \boldsymbol{J}\boldsymbol{v}_m - (r+\lambda)\boldsymbol{v}_m \quad \text{otherwise.} \end{aligned} \tag{20}$$

For American options, we obtain the following discrete LCPs:

$$\begin{aligned} &\left(\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_m}{\Delta\tau} - \boldsymbol{L}_{3a}\boldsymbol{v}_m\right) \geq \boldsymbol{0}, \quad \boldsymbol{v}_m \geq \boldsymbol{g}, \\ &\left(\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_m}{\Delta\tau} - \boldsymbol{L}_{3a}\boldsymbol{v}_m\right)^{\mathrm{T}} (\boldsymbol{v}_m - \boldsymbol{g}) = 0 \end{aligned} \tag{21}$$

for $m = 1$ and

$$\begin{aligned} &\left(\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_{m-1}}{2\Delta\tau} - \boldsymbol{L}_{3b}\boldsymbol{v}_m\right) \geq \boldsymbol{0}, \quad \boldsymbol{v}_m \geq \boldsymbol{g}, \\ &\left(\frac{\boldsymbol{v}_{m+1}-\boldsymbol{v}_{m-1}}{2\Delta\tau} - \boldsymbol{L}_{3b}\boldsymbol{v}_m\right)^{\mathrm{T}} (\boldsymbol{v}_m - \boldsymbol{g}) = 0 \end{aligned} \tag{22}$$

otherwise, where $\boldsymbol{L}_{3a}\boldsymbol{v}_m$ and $\boldsymbol{L}_{3b}\boldsymbol{v}_m$ are the right-hand sides of the equations in Equation (20), respectively. In [36], it was shown that the discretization is stable if $(r+2\lambda)\Delta\tau < \frac{1}{2}$.

5. Solution methods for American options under jump diffusion

We describe and compare three different methods for solving the LCPs arising from the pricing of American options. Then, we combine these methods with time discretizations employing either two or three time levels. Thus, we have a total of six different approaches.

As stated previously, the straightforward implicit discretization of the integral operator leads to a full matrix. We consider two approaches that solve the LCPs efficiently and maintain second-order accuracy. First, we employ an iterative method described in [50], which iterates a sequence of LCPs with a tridiagonal matrix when the two-level time discretization is used. Second, we use the three-level time discretization. This discretization leads to LCPs with tridiagonal matrices.

We combine these approaches with three methods for solving the resulting LCPs, which are the Brennan and Schwartz algorithm, the penalty method, and the operator splitting method. The Brennan–Schwartz algorithm and the operator splitting method are direct methods, while the penalty method is iterative. The operator splitting method and the penalty method give an approximation for the LCPs, while the Brennan–Schwartz algorithm solves the LCPs exactly. In the following, we denote the LCP

$$\boldsymbol{B}\boldsymbol{v} \geq \boldsymbol{b}, \quad \boldsymbol{v} \geq \boldsymbol{g}, \quad (\boldsymbol{B}\boldsymbol{v} - \boldsymbol{b})^{\mathrm{T}}(\boldsymbol{v} - \boldsymbol{g}) = 0 \tag{23}$$

by LCP$(\boldsymbol{B}, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{g})$.

5.1 *Iterative method with the Brennan and Schwartz algorithm*

With the two-level time discretization, following [50], we apply the iteration

$$\text{LCP}\,(\boldsymbol{T}, \boldsymbol{v}^{l+1}, \theta\boldsymbol{J}\boldsymbol{v}^{l} + \boldsymbol{b}, \boldsymbol{g}), \quad l = 0, 1, \ldots, \tag{24}$$

to compute the approximation $\boldsymbol{v}^{l+1}$ for $\boldsymbol{v}_{m+1}$, where $\boldsymbol{v}^0 = \boldsymbol{v}_m$, $\boldsymbol{T} = \boldsymbol{I}/\Delta\tau - \theta[\boldsymbol{D} - (r+\lambda)\boldsymbol{I}]$, and $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + (1-\theta)[\boldsymbol{D} - (r+\lambda)\boldsymbol{I} + \boldsymbol{J}]\boldsymbol{v}_m$. Here, $\boldsymbol{I}$ denotes the $N \times N$ identity matrix. Note that $\boldsymbol{T}$ is a tridiagonal matrix and the Brennan–Schwartz algorithm, given in Algorithm 2, can be used to solve the above LCPs. Furthermore, the special techniques discussed earlier can be applied on the dense multiplication $\theta\boldsymbol{J}\boldsymbol{v}_m$. It was shown in [50] that this iteration converges to the solution of the LCP (19) and that the rate of convergence is

$$C = \frac{\theta\Delta\tau\lambda}{1 + \theta\Delta\tau(r+\lambda)}. \tag{25}$$

In practice, $C \ll 1$ and, thus, convergence is very rapid. Typically, only a few iterations are necessary with a moderate jump activity λ. However, with a high jump activity, more iterations are required.

In the three-level time discretization, the discrete integral operator $\boldsymbol{J}$ is handled explicitly. Therefore, the matrices appearing in the LCPs (21) and (22) are tridiagonal and Algorithm 2 can be used to solve the LCPs directly.

The Brennan and Schwartz algorithm [8] is a modification of the Gaussian elimination technique [29]. We consider an LCP$(\boldsymbol{B}, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{g})$ with a tridiagonal $\boldsymbol{B}$. Form a $\boldsymbol{U}\boldsymbol{L}$ decomposition of $\boldsymbol{B}$ such that

$$\boldsymbol{U}\boldsymbol{L} = \boldsymbol{B}, \tag{26}$$

where $\boldsymbol{U}$ is an upper triangular matrix and $\boldsymbol{L}$ is a lower triangular matrix with ones on its diagonal. The algorithm is very fast, but it requires that the early exercise region be connected to the left boundary. This property can be guaranteed for put options when $\boldsymbol{B}$ is an M-matrix [1]. The pseudocode of the iterative method and the approach with three time level is given in Algorithm 1.

Algorithm 1: Iterative method with the Brennan–Schwartz algorithm.

begin
  for $m = 1, \ldots, M-1$ **do**
    if *time discretization not created or needs to be changed* **then**
      $\boldsymbol{T} = implicit\_tridiagonal\_part(\Delta\tau, \theta, m, \boldsymbol{D}, three\_levels)$
      $[\boldsymbol{U}, \boldsymbol{L}] = ul(\boldsymbol{T})$
    if *three_levels* **then**
      if *m=1* **then**
        $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + \boldsymbol{J}\boldsymbol{v}_m$
      else
        $\boldsymbol{b} = \boldsymbol{v}_{m-1}/2\Delta\tau + \boldsymbol{D}(\boldsymbol{v}_{m-1}/2) - (r+\lambda)\boldsymbol{v}_m + \boldsymbol{J}\boldsymbol{v}_m$
      $\boldsymbol{v}_{m+1} = bsalg(\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{b}, \boldsymbol{g})$ `// Call Algorithm 2`
    else
      $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + (1-\theta)(\boldsymbol{D}\boldsymbol{v}_m + \boldsymbol{J}\boldsymbol{v}_m)$
      `// Use iterative method`
      $l = 0, \quad \boldsymbol{v}^l = \boldsymbol{v}_m$
      repeat
        $\boldsymbol{v}^{l+1} = bsalg(\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{b} + \theta\boldsymbol{J}\boldsymbol{v}^l, \boldsymbol{g})$
        $l = l + 1$
      until $\|\boldsymbol{v}^l - \boldsymbol{v}^{l-1}\|_2 < 10^{-8}\|\boldsymbol{b}\|_2$
      $\boldsymbol{v}_{m+1} = \boldsymbol{v}^l$

Algorithm 2: The Brennan–Schwartz algorithm.

input : A $\boldsymbol{UL}$ decomposition of the tridiagonal matrix $\boldsymbol{B}$, vectors $\boldsymbol{b}, \boldsymbol{g}$
output: Solution vector $\boldsymbol{v}$ of LCP($\boldsymbol{B}, \boldsymbol{v}, \boldsymbol{b}, \boldsymbol{g}$).
function $\boldsymbol{v} = bsalg(\boldsymbol{U}, \boldsymbol{L}, \boldsymbol{b}, \boldsymbol{g})$
begin
  $\boldsymbol{y} = \boldsymbol{U}^{-1}\boldsymbol{b}$
  $v_1 = \max\{y_1, g_1\}$
  for $i = 2, \ldots, N$ **do**
    $v_i = \max\{y_i - L_{i,i-1}v_{i-1}, g_i\}$

5.2 *Operator splitting method*

The operator splitting approach for American option pricing was proposed by Ikonen and Toivanen [28]. It is based on an alternative formulation of the LCP (9) with a Lagrange multiplier Ψ as follows:

$$\begin{aligned} &(v_\tau - Lv) = \Psi, \\ &\Psi \geq 0, \quad v \geq g, \quad \Psi(v-g) = 0. \end{aligned} \tag{27}$$

The operator splitting method approximates the LCP and, thus, it introduces additional error. It was shown in [30] that the Crank–Nicolson discretization and the operator splitting method based on it have the same order of accuracy.

For the two-level time discretization, the operator splitting method reads

$$\begin{aligned}&\frac{\tilde{\boldsymbol{v}}_{m+1}-\boldsymbol{v}_m}{\Delta\tau}-\boldsymbol{D}(\theta\tilde{\boldsymbol{v}}_{m+1}+(1-\theta)\boldsymbol{v}_m)-\boldsymbol{J}(\theta\tilde{\boldsymbol{v}}_{m+1}+(1-\theta)\boldsymbol{v}_m)\\&\quad+(r+\lambda)(\theta\tilde{\boldsymbol{v}}_{m+1}+(1-\theta)\boldsymbol{v}_m)=\boldsymbol{\Psi}_m,\end{aligned}\tag{28}$$

$$\begin{aligned}&\frac{\boldsymbol{v}_{m+1}-\tilde{\boldsymbol{v}}_{m+1}}{\Delta\tau}=\boldsymbol{\Psi}_{m+1}-\boldsymbol{\Psi}_m,\\&\boldsymbol{\Psi}_{m+1}\geq 0,\quad \boldsymbol{v}_{m+1}\geq\boldsymbol{g},\quad (\boldsymbol{\Psi}_{m+1})^{\mathrm{T}}(\boldsymbol{v}_{m+1}-\boldsymbol{g})=0,\end{aligned}\tag{29}$$

where $\tilde{\boldsymbol{v}}_{m+1}$ is an intermediate solution vector and $\boldsymbol{\Psi}_m$ is the discrete Lagrange multiplier.

Each time step is split into two parts. First, in order to obtain the intermediate solution vector $\tilde{\boldsymbol{v}}_{m+1}$, the modified system of linear equations (28) is solved. Second, the intermediate solution $\tilde{\boldsymbol{v}}_{m+1}$ is projected to be feasible, and the Lagrange multiplier $\boldsymbol{\Psi}_m$ is updated to satisfy Equation (29). This can be performed componentwise following the update rule described by the pseudocode given in Algorithm 4.

The computational cost of the operator splitting method is dominated by the first step, where a system of linear equations needs to be solved. We can employ an iteration proposed by Tavella and Randall [51] to solve the system with a full matrix. This iteration reads

$$\boldsymbol{v}^{l+1}=\boldsymbol{T}^{-1}(\theta\boldsymbol{J}\boldsymbol{v}^l+\boldsymbol{b}),\quad l=0,1,\ldots,\tag{30}$$

where $\boldsymbol{v}^0$, $\boldsymbol{T}$, and $\boldsymbol{b}$ are defined in a way the same as for iteration (24).

In the case of the three-level time discretization, we employ the operator splitting method formulated in [37], which reads

$$\frac{\tilde{\boldsymbol{v}}_{m+1}-\boldsymbol{v}_m}{\Delta\tau}-\boldsymbol{D}\tilde{\boldsymbol{v}}_{m+1}-\boldsymbol{J}\boldsymbol{v}_m+(r+\lambda)\boldsymbol{v}_m=\boldsymbol{\Psi}_m,\tag{31}$$

$$\begin{aligned}&\frac{\boldsymbol{v}_{m+1}-\tilde{\boldsymbol{v}}_{m+1}}{\Delta\tau}=\boldsymbol{\Psi}_{m+1}-\boldsymbol{\Psi}_m,\\&\boldsymbol{\Psi}_{m+1}\geq\boldsymbol{0},\quad \boldsymbol{v}_{m+1}\geq\boldsymbol{g},\quad (\boldsymbol{\Psi}_{m+1})^{\mathrm{T}}(\boldsymbol{v}_{m+1}-\boldsymbol{g})=0\end{aligned}\tag{32}$$

for $m=1$ and

$$\frac{\tilde{\boldsymbol{v}}_{m+1}-\boldsymbol{v}_{m-1}}{2\Delta\tau}-\boldsymbol{D}\left(\frac{\tilde{\boldsymbol{v}}_{m+1}+\boldsymbol{v}_{m-1}}{2}\right)-\boldsymbol{J}\boldsymbol{v}_m+(r+\lambda)\boldsymbol{v}_m=\boldsymbol{\Psi}_m,\tag{33}$$

$$\begin{aligned}&\frac{\boldsymbol{v}_{m+1}-\tilde{\boldsymbol{v}}_{m+1}}{2\Delta\tau}=\boldsymbol{\Psi}_{m+1}-\boldsymbol{\Psi}_m,\\&\boldsymbol{\Psi}_{m+1}\geq\boldsymbol{0},\quad \boldsymbol{v}_{m+1}\geq\boldsymbol{g},\quad (\boldsymbol{\Psi}_{m+1})^{\mathrm{T}}(\boldsymbol{v}_{m+1}-\boldsymbol{g})=0\end{aligned}\tag{34}$$

otherwise.

In the first step, the system of linear equations in Equations (31) and (33) is tridiagonal and can be solved efficiently with an $\boldsymbol{LU}$ decomposition. The update step is the same as the one used in the case of the two-level time discretization with the modification that $2\Delta\tau$ is used instead of $\Delta\tau$ for $m>1$. The pseudocode for this approach is given in Algorithm 3.

Algorithm 3: Operator splitting method.

begin
  $\Psi = 0$
  for $m = 1, \ldots, M-1$ **do**
    if *time discretization not created or needs to be changed* **then**
      $\boldsymbol{T} = implicit\_tridiagonal\_part(\Delta\tau, \theta, m, \boldsymbol{D}, three\_levels)$
      $[\boldsymbol{L}, \boldsymbol{U}] = lu(\boldsymbol{T})$
    if *three_levels* **then**
      if *m=1* **then**
        $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + \boldsymbol{J}\boldsymbol{v}_m + \Psi$
      else
        $\boldsymbol{b} = \boldsymbol{v}_{m-1}/2\Delta\tau + \boldsymbol{D}(\boldsymbol{v}_{m-1}/2) - (r+\lambda)\boldsymbol{v}_m + \boldsymbol{J}\boldsymbol{v}_m + \Psi$
      $\boldsymbol{v}_{m+1} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}\boldsymbol{b}$
      $[\boldsymbol{v}_{m+1}, \Psi] = update(2\Delta\tau, \boldsymbol{g}, \boldsymbol{v}_{m+1}, \Psi)$ `// Call Algorithm 4`
    else
      $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + (1-\theta)(\boldsymbol{D}\boldsymbol{v}_m + \boldsymbol{J}\boldsymbol{v}_m) + \Psi$
      $l = 0, \quad \boldsymbol{v}^l = \boldsymbol{v}_m$
      repeat
        $\boldsymbol{v}^{l+1} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}(\boldsymbol{b} + \theta\boldsymbol{J}\boldsymbol{v}^l)$
        $[\boldsymbol{v}^{l+1}, \Psi] = update(\Delta\tau, \boldsymbol{g}, \boldsymbol{v}^{l+1}, \Psi)$
        $l = l + 1$
      until $\|\boldsymbol{v}^l - \boldsymbol{v}^{l-1}\|_2 < 10^{-8}\|\boldsymbol{b}\|_2$
      $\boldsymbol{v}_{m+1} = \boldsymbol{v}^l$

Algorithm 4: Update algorithm for the operator splitting method.

input : $\Delta\tau$ and vectors $\boldsymbol{g}, \boldsymbol{v}, \Psi$
output: Updated vectors $\boldsymbol{v}$ and Ψ.
function $[\boldsymbol{v}, \Psi] = update(\Delta\tau, \boldsymbol{g}, \boldsymbol{v}, \Psi)$
begin
  for $n = 1$ **to** N **do**
    if $v_n - \Delta\tau\Psi_n - g_n > 0$ **then**
      $v_n = v_n - \Delta\tau\Psi_n$
      $\Psi_n = 0$
    else
      $\Psi_n = \Psi_n + (g_n - v_n)/\Delta\tau$
      $v_n = g_n$

5.3 *Penalty method*

The considered penalty method for jump-diffusion models was proposed by d'Halluin *et al.* [23]. They approximated the LCP (9) by adding a penalty term for the violation of the early exercise

constraint into the PIDE (5) as follows:

$$v_\tau = Lv + \frac{1}{\epsilon}\max\{g - v, 0\}, \tag{35}$$

where ϵ is a small positive penalty parameter. This formulation leads to a nonlinear and non-smooth discretized system of equations.

A common approach to solve such a system is to employ a semismooth Newton method; see [32,61], for example. Let $\boldsymbol{P}^l$ be a diagonal matrix defined by

$$P^l_{n,n} = \begin{cases} \dfrac{1}{\epsilon} & \text{if } v^l_n < g_n, \\ 0 & \text{otherwise,} \end{cases} \quad n = 1, \ldots, N. \tag{36}$$

In the case of the three-level time discretization, we use the semismooth Newton iteration

$$\boldsymbol{v}^{l+1} = (\boldsymbol{C}^l)^{-1}(\boldsymbol{b} + \boldsymbol{P}^l\boldsymbol{g}), \quad l = 0, 1, 2, \ldots, \tag{37}$$

where $\boldsymbol{C}^l$ is a generalized Jacobian matrix given by

$$\boldsymbol{C}^l = \frac{1}{\Delta\tau}\boldsymbol{I} - \boldsymbol{D} + (r + \lambda)\boldsymbol{I} + \boldsymbol{P}^l \quad \text{and} \quad \boldsymbol{b} = \boldsymbol{J}\boldsymbol{v}_m \quad \text{for } m = 1, \tag{38}$$

and

$$\begin{aligned} \boldsymbol{C}^l &= \frac{1}{2\Delta\tau}\boldsymbol{I} - \frac{1}{2}\boldsymbol{D} + \boldsymbol{P}^l \quad \text{and} \\ \boldsymbol{b} &= \left[\frac{1}{2\Delta\tau}\boldsymbol{I} + \frac{1}{2}\boldsymbol{D}\right]\boldsymbol{v}_{m-1} + [-(r + \lambda)\boldsymbol{I} + \boldsymbol{J}]\boldsymbol{v}_m \quad \text{otherwise.} \end{aligned} \tag{39}$$

Note that the integral operator does not appear in the iteration. Thus, the resulting systems with $\boldsymbol{C}^l$ can be solved efficiently with an $\boldsymbol{LU}$ decomposition.

In the case of the two-level time discretization, a generalized Jacobian is a full matrix. Thus, the above semismooth Newton iteration would require solving linear problems with a full matrix. Instead, we employ an approximate semismooth Newton method described in [23,52]. We use a tridiagonal approximation $\boldsymbol{C}^l$ without the discrete integral operator for the generalized Jacobian matrix. The approximate semismooth Newton iteration reads

$$\boldsymbol{v}^{l+1} = (\boldsymbol{C}^l)^{-1}(\boldsymbol{b} + \theta\boldsymbol{J}\boldsymbol{v}^l + \boldsymbol{P}^l\boldsymbol{g}), \quad l = 0, 1, 2, \ldots, \tag{40}$$

which has been shown in [23] to rapidly converge to the unique solution of Equation (35). In the above, $\boldsymbol{C}^l$ and $\boldsymbol{b}$ are given by

$$\begin{aligned} \boldsymbol{C}^l &= \frac{1}{\Delta\tau}\boldsymbol{I} - \theta[\boldsymbol{D} - (r + \lambda)\boldsymbol{I}] + \boldsymbol{P}^l \quad \text{and} \\ \boldsymbol{b} &= \frac{1}{\Delta\tau}\boldsymbol{v}_m + (1 - \theta)[\boldsymbol{D} - (r + \lambda)\boldsymbol{I} + \boldsymbol{J}]\boldsymbol{v}_m. \end{aligned} \tag{41}$$

The pseudocode for the penalty method is given in Algorithm 5.

Algorithm 5: Penalty method.

begin
  for $m = 1, \ldots, M-1$ **do**
    if *time discretization not created or needs to be changed* **then**
      $\boldsymbol{T} = implicit\_tridiagonal\_part(\Delta\tau, \theta, m, \boldsymbol{D}, three\_levels)$
      $[\boldsymbol{L}, \boldsymbol{U}] = lu(\boldsymbol{T})$
    if *three_levels* **then**
      if *m=1* **then**
        $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + \boldsymbol{J}\boldsymbol{v}_m$
      else
        $\boldsymbol{b} = \boldsymbol{v}_{m-1}/2\Delta\tau + \boldsymbol{D}(\boldsymbol{v}_{m-1}/2) - (r+\lambda)\boldsymbol{v}_m + \boldsymbol{J}\boldsymbol{v}_m$
      $l = 0, \quad \boldsymbol{v}^l = \boldsymbol{v}_m$
      repeat
        for $n = 1, \ldots, N$ **do**
          if $v_n^l < g_n$ **then**
            $P_{n,n}^l = 1/\epsilon$
          else
            $P_{n,n}^l = 0$
        $\boldsymbol{v}^{l+1} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}(\boldsymbol{b} + \boldsymbol{P}^l\boldsymbol{g})$
        $l = l + 1$
      until $\|\boldsymbol{v}^l - \boldsymbol{v}^{l-1}\|_2 < 10^{-8}\|\boldsymbol{b}\|_2$
      $\boldsymbol{v}_{m+1} = \boldsymbol{v}^l$
    else
      $\boldsymbol{b} = \boldsymbol{v}_m/\Delta\tau + (1-\theta)(\boldsymbol{D}\boldsymbol{v}_m + \boldsymbol{J}\boldsymbol{v}_m)$
      $l = 0, \quad \boldsymbol{v}^l = \boldsymbol{v}_m$
      repeat
        for $n = 1, \ldots, N$ **do**
          if $v_n^l < g_n$ **then**
            $P_{n,n}^l = 1/\epsilon$
          else
            $P_{n,n}^l = 0$
        $\boldsymbol{v}^{l+1} = \boldsymbol{U}^{-1}\boldsymbol{L}^{-1}(\boldsymbol{b} + \theta\boldsymbol{J}\boldsymbol{v}^l + \boldsymbol{P}^l\boldsymbol{g})$
        $l = l + 1$
      until $\|\boldsymbol{v}^l - \boldsymbol{v}^{l-1}\|_2 < 10^{-8}\|\boldsymbol{b}\|_2$
      $\boldsymbol{v}_{m+1} = \boldsymbol{v}^l$

Table 1. Reference prices used in numerical experiments.

Model and type	Value at 90	Value at 100	Value at 110
Kou, American put [23,52]	10.005071	2.807879	0.561876
Merton, American put [24,37]	10.003822	3.241251	1.419803
Kou, American put (second parameter set)	10.698208	6.417275	4.624099
Merton, American put (second parameter set)	19.948906	18.246332	16.666925

Table 2. Pricing errors for an American put option under Kou's model with the first parameter set.

			Two time levels, error at			Three time levels, error at		
Method	N	M	$x = 90$	$x = 100$	$x = 110$	$x = 90$	$x = 100$	$x = 110$
Iterative method with the Brennan–Schwartz algorithm	51	20	1.504×10^{-1}	-8.030×10^{-3}	1.047×10^{-1}	1.504×10^{-1}	-7.550×10^{-3}	1.046×10^{-1}
	101	40	2.114×10^{-2}	-1.201×10^{-1}	-3.488×10^{-2}	2.112×10^{-2}	-1.192×10^{-1}	-3.494×10^{-2}
	201	80	-5.071×10^{-3}	-3.000×10^{-2}	-5.804×10^{-3}	-5.071×10^{-3}	-2.983×10^{-2}	-5.820×10^{-3}
	401	160	-4.264×10^{-3}	-7.636×10^{-3}	-1.552×10^{-3}	-4.275×10^{-3}	-7.541×10^{-3}	-1.538×10^{-3}
	801	320	-3.127×10^{-4}	-1.970×10^{-3}	-4.153×10^{-4}	-3.120×10^{-4}	-1.939×10^{-3}	-4.089×10^{-4}
	1601	640	-1.006×10^{-4}	-5.141×10^{-4}	-1.127×10^{-4}	-1.005×10^{-4}	-5.075×10^{-4}	-1.116×10^{-4}
Operator method	51	20	1.504×10^{-1}	-8.030×10^{-3}	1.047×10^{-1}	1.504×10^{-1}	-7.666×10^{-3}	1.046×10^{-1}
	101	40	2.114×10^{-2}	-1.201×10^{-1}	-3.488×10^{-2}	2.108×10^{-2}	-1.195×10^{-1}	-3.502×10^{-2}
	201	80	-5.071×10^{-3}	-2.999×10^{-2}	-5.804×10^{-3}	-5.071×10^{-3}	-2.971×10^{-2}	-5.812×10^{-3}
	401	160	-4.263×10^{-3}	-7.635×10^{-3}	-1.552×10^{-3}	-4.316×10^{-3}	-7.536×10^{-3}	-1.557×10^{-3}
	801	320	-3.077×10^{-4}	-1.969×10^{-3}	-4.150×10^{-4}	-1.925×10^{-4}	-1.916×10^{-3}	-4.151×10^{-4}
	1601	640	-9.695×10^{-5}	-5.113×10^{-4}	-1.122×10^{-4}	-4.188×10^{-5}	-4.727×10^{-4}	-1.076×10^{-4}
Penalty method	51	20	1.504×10^{-1}	-8.030×10^{-3}	1.047×10^{-1}	1.504×10^{-1}	-7.550×10^{-3}	1.046×10^{-1}
	101	40	2.114×10^{-2}	-1.201×10^{-1}	-3.488×10^{-2}	2.112×10^{-2}	-1.192×10^{-1}	-3.494×10^{-2}
	201	80	-5.071×10^{-3}	-3.000×10^{-2}	-5.804×10^{-3}	-5.072×10^{-3}	-2.983×10^{-2}	-5.820×10^{-3}
	401	160	-4.264×10^{-3}	-7.636×10^{-3}	-1.552×10^{-3}	-4.276×10^{-3}	-7.541×10^{-3}	-1.538×10^{-3}
	801	320	-3.129×10^{-4}	-1.970×10^{-3}	-4.153×10^{-4}	-3.123×10^{-4}	-1.939×10^{-3}	-4.089×10^{-4}
	1601	640	-1.007×10^{-4}	-5.141×10^{-4}	-1.127×10^{-4}	-1.007×10^{-4}	-5.076×10^{-4}	-1.116×10^{-4}

6. Numerical experiments

This section presents numerical results computed with the selection of methods described earlier for pricing American options under jump diffusion. The first set of experiments is performed under parameters that are the same as those used in [23,24,37,52], which are given by

$$\sigma = 0.15, \quad r = 0.05, \quad T = 0.25, \quad K = 100, \quad \text{and} \quad \lambda = 0.1. \tag{42}$$

The second set of parameters is chosen as

$$\sigma = 0.1, \quad r = 0.1, \quad T = 1, \quad K = 100, \quad \text{and} \quad \lambda = 0.5, \tag{43}$$

where we have noticeably longer maturity T and higher jump intensity λ. The jump distribution parameters are not changed between experiments. For Kou's model, the jump parameters are given by

$$\alpha_1 = 3.0465, \quad \alpha_2 = 3.0775, \quad \text{and} \quad p = 0.3445, \tag{44}$$

and for Merton's model, they are given by

$$\gamma = -0.9 \quad \text{and} \quad \delta = 0.45. \tag{45}$$

A uniform grid with N nodes is used and the truncation boundary is set to $X = 400$. The number of time steps is M. The parameter θ for the Rannacher scheme is set to $\theta = 1$ (the implicit Euler method) for the first time step $m = 1$ and $\theta = 0.5$ (the Crank–Nicolson method) for $m > 1$. The stopping criterion for the iterations (24), (30), (37), and (40) is chosen as $\|\boldsymbol{v}^{l+1} - \boldsymbol{v}^l\|_2 < 10^{-8}\|\boldsymbol{b}\|_2$. The penalty parameter for the penalty method is set to $\epsilon = 10^{-4}$.

We present the numerical results for American put options under Kou's and Merton's model. The reference prices for these options are given in Table 1. For the second set of parameters, we computed the reference prices numerically with a fine grid of $N = 25,601$, $M = 10,240$. The computations were performed on a PC with a 2.3 GHz Intel Xeon E5410 processor. The code

Table 3. CPU times, total iteration counts, and ratios of convergence for an American put option under Kou's model with the first parameter set.

			Two time levels			Three time levels		
Method	N	M	Ratio	Iterations	Time (ms)	Ratio	Iterations	Time (ms)
Iterative method	51	20		60	0.3		–	0.2
with the Brennan–	101	40	1.45	120	0.4	1.46	–	0.2
Schwartz algorithm	201	80	4.09	191	1.1	4.09	–	0.6
	401	160	3.49	320	3.4	3.50	–	1.8
	801	320	4.36	640	11.7	4.39	–	6.4
	1601	640	3.80	1280	42.4	3.79	–	20.9
Operator splitting	51	20		61	0.3		–	0.2
method	101	40	1.45	122	0.4	1.45	–	0.2
	201	80	4.09	244	1.3	4.11	–	0.6
	401	160	3.49	469	4.6	3.48	–	1.8
	801	320	4.36	804	13.8	4.48	–	6.4
	1601	640	3.82	1404	46.7	4.05	–	21.5
Penalty method	51	20		61	0.2		41	0.2
	101	40	1.45	121	0.4	1.46	83	0.3
	201	80	4.09	193	1.1	4.09	164	0.8
	401	160	3.49	330	3.5	3.50	330	2.8
	801	320	4.36	658	11.5	4.39	657	9.4
	1601	640	3.80	1314	43.0	3.79	1318	34.3

Table 4. Pricing errors for an American put option under Merton's model with the first parameter set.

			Two time levels, error at			Three time levels, error at		
Method	N	M	$x = 90$	$x = 100$	$x = 110$	$x = 90$	$x = 100$	$x = 110$
Iterative method	51	20	1.758×10^{-1}	-1.028×10^{-1}	4.888×10^{-2}	1.758×10^{-1}	-1.026×10^{-1}	4.900×10^{-2}
with the Brennan–	101	40	2.505×10^{-2}	-1.295×10^{-1}	-3.598×10^{-2}	2.510×10^{-2}	-1.286×10^{-1}	-3.596×10^{-2}
Schwartz algorithm	201	80	-3.822×10^{-3}	-3.254×10^{-2}	-6.438×10^{-3}	-3.822×10^{-3}	-3.226×10^{-2}	-6.410×10^{-3}
	401	160	-3.822×10^{-3}	-8.266×10^{-3}	-1.693×10^{-3}	-3.822×10^{-3}	-8.183×10^{-3}	-1.681×10^{-3}
	801	320	-8.635×10^{-4}	-2.128×10^{-3}	-4.459×10^{-4}	-8.662×10^{-4}	-2.105×10^{-3}	-4.424×10^{-4}
	1601	640	-2.916×10^{-4}	-5.517×10^{-4}	-1.177×10^{-4}	-2.907×10^{-4}	-5.464×10^{-4}	-1.169×10^{-4}
Operator splitting	51	20	1.758×10^{-1}	-1.028×10^{-1}	4.888×10^{-2}	1.758×10^{-1}	-1.027×10^{-1}	4.898×10^{-2}
method	101	40	2.505×10^{-2}	-1.295×10^{-1}	-3.598×10^{-2}	2.477×10^{-2}	-1.293×10^{-1}	-3.607×10^{-2}
	201	80	-3.822×10^{-3}	-3.253×10^{-2}	-6.438×10^{-3}	-3.822×10^{-3}	-3.224×10^{-2}	-6.410×10^{-3}
	401	160	-3.822×10^{-3}	-8.265×10^{-3}	-1.693×10^{-3}	-3.822×10^{-3}	-8.163×10^{-3}	-1.697×10^{-3}
	801	320	-8.583×10^{-4}	-2.127×10^{-3}	-4.458×10^{-4}	-7.270×10^{-4}	-2.052×10^{-3}	-4.418×10^{-4}
	1601	640	-2.875×10^{-4}	-5.492×10^{-4}	-1.173×10^{-4}	-1.955×10^{-4}	-5.036×10^{-4}	-1.125×10^{-4}
Penalty method	51	20	1.758×10^{-1}	-1.029×10^{-1}	4.888×10^{-2}	1.758×10^{-1}	-1.026×10^{-1}	4.900×10^{-2}
	101	40	2.505×10^{-2}	-1.295×10^{-1}	-3.598×10^{-2}	2.509×10^{-2}	-1.286×10^{-1}	-3.596×10^{-2}
	201	80	-3.822×10^{-3}	-3.254×10^{-2}	-6.438×10^{-3}	-3.823×10^{-3}	-3.226×10^{-2}	-6.410×10^{-3}
	401	160	-3.822×10^{-3}	-8.266×10^{-3}	-1.693×10^{-3}	-3.822×10^{-3}	-8.183×10^{-3}	-1.681×10^{-3}
	801	320	-8.638×10^{-4}	-2.128×10^{-3}	-4.460×10^{-4}	-8.668×10^{-4}	-2.105×10^{-3}	-4.424×10^{-4}
	1601	640	-2.916×10^{-4}	-5.518×10^{-4}	-1.177×10^{-4}	-2.908×10^{-4}	-5.464×10^{-4}	-1.169×10^{-4}

Table 5. CPU times, total iteration counts, and ratios of convergence for an American put option under Merton's model with the first parameter set.

			Two time levels			Three time levels		
Method	N	M	Ratio	Iterations	Time (ms)	Ratio	Iterations	Time (ms)
Iterative method with the Brennan–Schwartz algorithm	51	20		60	0.9		–	0.3
	101	40	1.53	120	2.5	1.54	–	0.8
	201	80	4.10	240	9.5	4.10	–	2.7
	401	160	3.60	321	27.5	3.60	–	10.6
	801	320	3.96	640	115.7	3.96	–	40.2
	1601	640	3.68	1280	531.0	3.68	–	183.1
Operator splitting method	51	20		61	0.8		–	0.3
	101	40	1.53	123	2.6	1.53	–	0.8
	201	80	4.10	246	9.6	4.13	–	2.7
	401	160	3.60	488	37.7	3.61	–	10.3
	801	320	3.96	911	147.6	4.13	–	40.4
	1601	640	3.70	1471	579.0	4.02	–	183.6
Penalty method	51	20		61	0.8		41	0.4
	101	40	1.53	121	2.6	1.54	83	0.8
	201	80	4.10	241	8.8	4.10	165	2.9
	401	160	3.60	330	28.1	3.60	329	10.8
	801	320	3.96	658	117.1	3.96	659	43.2
	1601	640	3.68	1314	538.7	3.68	1316	197.1

was run on Matlab; however, the solution routine was compiled from a MEX-file written in the C language.

The pricing errors at $x = 90, 100$, and 110 for an American put option under Kou's model with the first parameter set are listed in Table 2 with different values of N and M. Total iteration counts, CPU times and the ratios of consecutive errors are reported in Table 3. The corresponding results for the American put option under Merton's model are given in Tables 4 and 5. The ratios of errors at $x = 90, 100$, and 110 are computed using the l_2-norm. The pricing errors for the second set of experiments are given in Tables 6 and 7, and the convergence information is given in Tables 8 and 9. The absolute pricing errors for $N = 1601$, $M = 640$ are also plotted in Figure 1.

The pricing errors are almost identical with all the different approaches. The approximation error of the penalty method is almost non-existent when compared with the Brennan–Schwartz algorithm, whereas the operator splitting method in some cases produces a visible, although small approximation error. The results show second-order convergence for all the approaches. The ratios of convergence behave somewhat erratically. This is mainly caused by the error at $x = 90$, which is close to the early exercise boundary. Another reason for this erratic behaviour between $N = 101$ and $N = 201$ is the use of quadratic interpolation between grid points when needed. The convergence is of second order, on average, but the actual improvement of the error at $x = 90$ for each refinement of the grid depends on how close the new grid point is to the early exercise boundary.

The iterative methods require two iterations per time step to converge with the first parameter set and from three to four iterations to converge with the second parameter set. For the two-level time discretization, the penalty method requires less iterations to converge than the operator splitting method, and it is, therefore, somewhat faster. The Brennan–Schwartz algorithm has more restrictive requirements on the payoff function, and it is only slightly faster than the other methods. Thus, we deduce that the penalty method and operator splitting method are efficient general-purpose methods.

The three-level time discretization is significantly faster and has accuracy similar to that of the iterative approaches based on the two-level time discretization. The performance difference is

Table 6. Pricing errors for an American put option under Kou's model with the second parameter set.

			Two time levels, error at			Three time levels, error at		
Method	N	M	$x = 90$	$x = 100$	$x = 110$	$x = 90$	$x = 100$	$x = 110$
Iterative method with	51	20	-1.496×10^{-1}	-2.610×10^{-1}	-1.604×10^{-2}	-1.525×10^{-1}	-2.566×10^{-1}	-1.144×10^{-2}
the Brennan–Schwartz	101	40	-1.435×10^{-1}	-7.561×10^{-2}	-2.502×10^{-2}	-1.426×10^{-1}	-7.276×10^{-2}	-2.379×10^{-2}
algorithm	201	80	-2.891×10^{-2}	-1.913×10^{-2}	-5.629×10^{-3}	-2.865×10^{-2}	-1.856×10^{-2}	-5.323×10^{-3}
	401	160	-7.274×10^{-3}	-4.914×10^{-3}	-1.475×10^{-3}	-7.196×10^{-3}	-4.742×10^{-3}	-1.392×10^{-3}
	801	320	-1.708×10^{-3}	-1.258×10^{-3}	-3.822×10^{-4}	-1.678×10^{-3}	-1.209×10^{-3}	-3.620×10^{-4}
	1601	640	-4.370×10^{-4}	-3.261×10^{-4}	-1.001×10^{-4}	-4.276×10^{-4}	-3.116×10^{-4}	-9.513×10^{-5}
Operator splitting method	51	20	-1.496×10^{-1}	-2.610×10^{-1}	-1.604×10^{-2}	-1.523×10^{-1}	-2.558×10^{-1}	-1.145×10^{-2}
	101	40	-1.435×10^{-1}	-7.561×10^{-2}	-2.502×10^{-2}	-1.468×10^{-1}	-7.368×10^{-2}	-2.403×10^{-2}
	201	80	-2.890×10^{-2}	-1.913×10^{-2}	-5.629×10^{-3}	-2.842×10^{-2}	-1.877×10^{-2}	-5.349×10^{-3}
	401	160	-7.272×10^{-3}	-4.913×10^{-3}	-1.475×10^{-3}	-6.514×10^{-3}	-4.678×10^{-3}	-1.374×10^{-3}
	801	320	-1.705×10^{-3}	-1.257×10^{-3}	-3.819×10^{-4}	-1.225×10^{-3}	-1.108×10^{-3}	-3.385×10^{-4}
	1601	640	-4.300×10^{-4}	-3.243×10^{-4}	-9.954×10^{-5}	-2.139×10^{-4}	-2.614×10^{-4}	-8.180×10^{-5}
Penalty method	51	20	-1.496×10^{-1}	-2.610×10^{-1}	-1.605×10^{-2}	-1.525×10^{-1}	-2.566×10^{-1}	-1.145×10^{-2}
	101	40	-1.435×10^{-1}	-7.562×10^{-2}	-2.502×10^{-2}	-1.426×10^{-1}	-7.277×10^{-2}	-2.380×10^{-2}
	201	80	-2.891×10^{-2}	-1.913×10^{-2}	-5.630×10^{-3}	-2.866×10^{-2}	-1.856×10^{-2}	-5.325×10^{-3}
	401	160	-7.275×10^{-3}	-4.915×10^{-3}	-1.476×10^{-3}	-7.199×10^{-3}	-4.743×10^{-3}	-1.393×10^{-3}
	801	320	-1.709×10^{-3}	-1.259×10^{-3}	-3.825×10^{-4}	-1.679×10^{-3}	-1.210×10^{-3}	-3.626×10^{-4}
	1601	640	-4.373×10^{-4}	-3.263×10^{-4}	-1.003×10^{-4}	-4.283×10^{-4}	-3.121×10^{-4}	-9.544×10^{-5}

Table 7. Pricing errors for an American put option under Merton's model with the second parameter set.

			Two time levels, error at			Three time levels, error at		
Method	N	M	$x = 90$	$x = 100$	$x = 110$	$x = 90$	$x = 100$	$x = 110$
Iterative method with the Brennan–Schwartz algorithm	51	20	9.288×10^{-2}	-2.852×10^{-3}	2.591×10^{-2}	1.100×10^{-1}	2.315×10^{-2}	5.069×10^{-2}
	101	40	-8.714×10^{-3}	-2.249×10^{-3}	-1.020×10^{-3}	-2.241×10^{-3}	4.427×10^{-3}	5.178×10^{-3}
	201	80	-1.123×10^{-2}	-5.063×10^{-3}	-4.767×10^{-3}	-9.534×10^{-3}	-3.387×10^{-3}	-3.217×10^{-3}
	401	160	-3.021×10^{-3}	-1.275×10^{-3}	-1.178×10^{-3}	-2.601×10^{-3}	-8.565×10^{-4}	-7.919×10^{-4}
	801	320	-7.809×10^{-4}	-3.298×10^{-4}	-3.027×10^{-4}	-6.757×10^{-4}	-2.250×10^{-4}	-2.060×10^{-4}
	1601	640	-1.988×10^{-4}	-8.358×10^{-5}	-7.650×10^{-5}	-1.724×10^{-4}	-5.780×10^{-5}	-5.289×10^{-5}
Operator splitting method	51	20	9.288×10^{-2}	-2.852×10^{-3}	2.591×10^{-2}	1.010×10^{-1}	2.315×10^{-2}	4.908×10^{-2}
	101	40	-8.714×10^{-3}	-2.249×10^{-3}	-1.020×10^{-3}	-2.331×10^{-3}	3.895×10^{-3}	4.499×10^{-3}
	201	80	-1.123×10^{-2}	-5.063×10^{-3}	-4.767×10^{-3}	-9.663×10^{-3}	-3.553×10^{-3}	-3.429×10^{-3}
	401	160	-3.021×10^{-3}	-1.275×10^{-3}	-1.178×10^{-3}	-2.632×10^{-3}	-8.733×10^{-4}	-8.126×10^{-4}
	801	320	-7.809×10^{-4}	-3.297×10^{-4}	-3.027×10^{-4}	-6.682×10^{-4}	-2.218×10^{-4}	-2.018×10^{-4}
	1601	640	-1.987×10^{-4}	-8.351×10^{-5}	-7.641×10^{-5}	-1.651×10^{-4}	-5.159×10^{-5}	-4.488×10^{-5}
Penalty method	51	20	9.286×10^{-2}	-2.876×10^{-3}	2.588×10^{-2}	1.100×10^{-1}	2.310×10^{-2}	5.065×10^{-2}
	101	40	-8.723×10^{-3}	-2.258×10^{-3}	-1.029×10^{-3}	-2.260×10^{-3}	4.409×10^{-3}	5.161×10^{-3}
	201	80	-1.123×10^{-2}	-5.067×10^{-3}	-4.771×10^{-3}	-9.543×10^{-3}	-3.396×10^{-3}	-3.225×10^{-3}
	401	160	-3.024×10^{-3}	-1.277×10^{-3}	-1.181×10^{-3}	-2.605×10^{-3}	-8.608×10^{-4}	-7.960×10^{-4}
	801	320	-7.820×10^{-4}	-3.309×10^{-4}	-3.037×10^{-4}	-6.779×10^{-4}	-2.271×10^{-4}	-2.080×10^{-4}
	1601	640	-1.994×10^{-4}	-8.412×10^{-5}	-7.701×10^{-5}	-1.735×10^{-4}	-5.887×10^{-5}	-5.392×10^{-5}

Table 8. CPU times, total iteration counts, and ratios of convergence for an American put option under Kou's model with the second parameter set.

			Two time levels			Three time levels		
Method	N	M	Ratio	Iterations	Time (ms)	Ratio	Iterations	Time (ms)
Iterative method	51	20		81	0.2		–	0.2
with the Brennan–	101	40	1.84	160	0.5	1.85	–	0.2
Schwartz algorithm	201	80	4.67	241	1.3	4.68	–	0.6
	401	160	3.95	480	4.6	3.96	–	1.8
	801	320	4.13	960	15.3	4.16	–	6.4
	1601	640	3.89	1920	58.7	3.90	–	21.2
Operator splitting	51	20		81	0.3		–	0.2
method	101	40	1.84	161	0.5	1.79	–	0.2
	201	80	4.67	320	1.6	4.82	–	0.6
	401	160	3.95	649	5.9	4.24	–	1.8
	801	320	4.13	1286	19.7	4.83	–	6.2
	1601	640	3.93	2242	68.7	4.85	–	21.4
Penalty method	51	20		82	0.2		41	0.2
	101	40	1.84	161	0.5	1.85	84	0.3
	201	80	4.67	242	1.3	4.68	166	0.8
	401	160	3.94	481	4.6	3.96	333	2.8
	801	320	4.13	961	15.3	4.16	666	9.5
	1601	640	3.89	1921	58.4	3.90	1329	34.6

Table 9. CPU times, total iteration counts, and ratios of convergence for an American put option under Merton's model with the second parameter set.

			Two time levels			Three time levels		
Method	N	M	Ratio	Iterations	Time (ms)	Ratio	Iterations	Time (ms)
Iterative method with	51	20		80	1.0		–	0.3
the Brennan–Schwartz	101	40	10.65	160	3.1	17.20	–	0.8
algorithm	201	80	0.69	241	9.2	0.68	–	2.7
	401	160	3.79	480	36.8	3.72	–	10.2
	801	320	3.87	960	154.0	3.84	–	40.0
	1601	640	3.93	1920	714.9	3.92	–	182.4
Operator splitting method	51	20		80	0.9		–	0.3
	101	40	10.65	160	3.1	17.94	–	0.8
	201	80	0.69	306	10.7	0.59	–	2.7
	401	160	3.79	754	52.8	3.76	–	10.2
	801	320	3.87	1658	240.2	3.95	–	40.1
	1601	640	3.94	3370	1120.7	4.10	–	183.2
Penalty method	51	20		84	1.0		43	0.3
	101	40	10.63	161	3.1	17.23	87	0.8
	201	80	0.69	253	9.7	0.67	172	3.0
	401	160	3.79	481	36.8	3.72	344	11.3
	801	320	3.87	961	154.1	3.84	688	43.7
	1601	640	3.93	1921	712.1	3.90	1375	196.1

more evident when pricing the American options under Merton's model. Under Merton's model, the dense matrix–vector multiplication requires $O(N \log N)$ operations instead of $O(N)$ operations in the case of Kou's model. The three-level time discretization only performs one multiplication by a full matrix per time step. Even in the case of the penalty method with the three-level time discretization, the integral operator does not appear in the penalty iteration. Thus, the iteration is very efficient. However, combining the Brennan–Schwartz algorithm or the operator splitting method with the three-level time discretization leads to a direct method, which is even faster. The

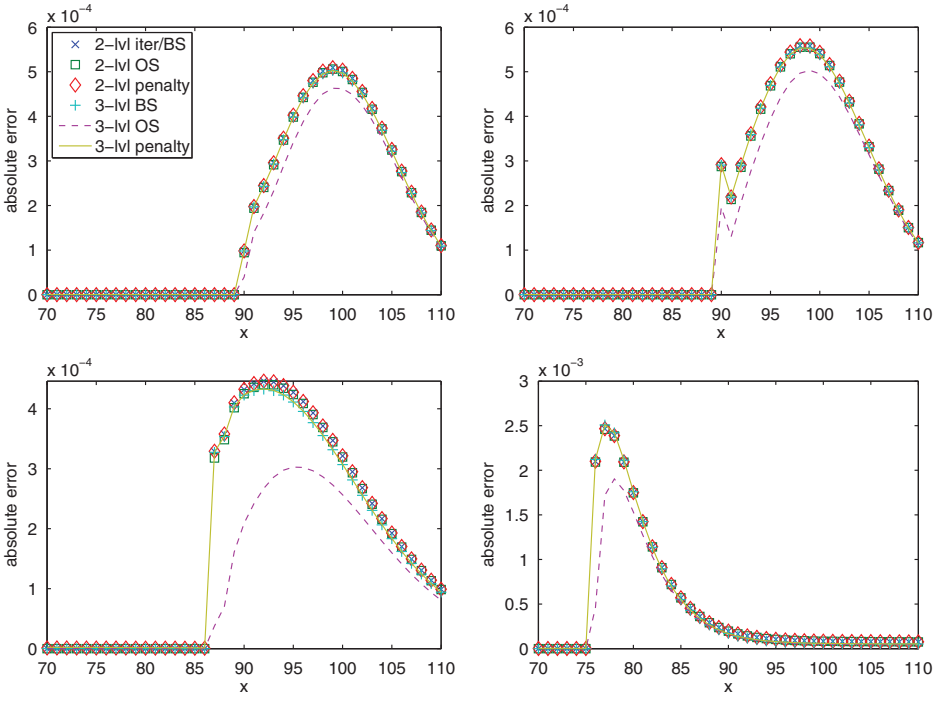

Figure 1. Absolute errors computed using the six different approaches with $N = 1601$, $M = 640$ from Table 2 (top left), Table 4 (top right), Table 6 (bottom left), and Table 7 (bottom right).

operator splitting method combined with the three-level time discretization leads to an efficient direct method for pricing American options under jump-diffusion models without the restrictions of the Brennan–Schwartz algorithm.

7. Conclusions

We have presented a survey on the research on PIDE formulations and numerical methods for American option pricing under jump-diffusion models. Several ways to treat the early exercise constraint have been described. The most common approach is to formulate an LCP. Finite difference discretizations are most often used for the differential part of the underlying operator. Many solution methods for the resulting system of equations have been discussed.

We performed numerical experiments with six different methods for pricing American options under finite activity jump-diffusion models. All methods are second-order accurate in time and space. Two of these methods are direct, while the other methods are iterative. The time discretization with two time levels leads to systems with full matrices, while the time discretization with three time levels leads to systems with tridiagonal matrices. Rapidly converging iterations have been used to solve the full systems efficiently. Each iteration requires a multiplication by a full matrix, which can be performed fast using an FFT. Still, the iterative methods cannot compete in performance with the direct methods.

Combining the time discretization with three time levels and the Brennan–Schwartz algorithm leads to a fast direct method which solves the LCP without additional approximations. The Brennan–Schwarz algorithm can be applied for put and call options, but it is not applicable for some other payoff functions. The penalty method is iterative, and consequently it is not as

efficient as the direct methods. However, the penalty method is generic as it can be used with any payoff function. For the considered American put options, the operator splitting method based on the discretization with three time levels was fast and accurate.

References

[1] Y. Achdou and O. Pironneau, *Computational methods for option pricing*, Frontiers in Applied Mathematics Vol. 30, SIAM, Philadelphia, PA, 2005.
[2] A. Almendral and C.W. Oosterlee, *Numerical valuation of options with jumps in the underlying*, Appl. Numer. Math. 53 (2005), pp. 1–18.
[3] L. Andersen and J. Andreasen, *Jump-diffusion processes: Volatility smile fitting and numerical methods for option pricing*, Rev. Deriv. Res. 4 (2000), pp. 231–262.
[4] D.S. Bates, *Jumps and stochastic volatility: Exchange rate processes implicit Deutsche mark options*, Rev. Financ. Stud. 9 (1996), pp. 69–107.
[5] E. Bayraktar and H. Xing, *Pricing American options for jump diffusions by iterating optimal stopping problems for diffusions*, Math. Methods Oper. Res. 70 (2009), pp. 505–525.
[6] F.E. Benth, K.H. Karlsen, and K. Reikvam, *A semilinear Black and Scholes partial differential equation for valuing American options: Approximate solutions and convergence*, Interfaces Free Bound. 6 (2004), pp. 379–404.
[7] F. Black and M. Scholes, *The pricing of options and corporate liabilities*, J. Political Econ. 81 (1973), pp. 637–654.
[8] M.J. Brennan and E.S. Schwartz, *The valuation of American put options*, J. Finance 32 (1977), pp. 449–462.
[9] M. Briani, R. Natalini, and G. Russo, *Implicit–explicit numerical schemes for jump-diffusion processes*, Calcolo 44 (2007), pp. 33–57.
[10] P. Carr and D.B. Madan, *Option valuation using the fast Fourier transform*, J. Comput. Finance 2 (1999), pp. 61–73.
[11] P. Carr and A. Mayo, *On the numerical evaluation of options prices in jump diffusion processes*, Eur. J. Finance 13 (2007), pp. 353–372.
[12] P. Carr, H. Geman, D.B. Madan, and M. Yor, *The fine structure of asset returns: An empirical investigation*, J. Bus. 75 (2002), pp. 305–332.
[13] R. Cont and P. Tankov, *Financial modelling with jump processes*, Chapman & Hall, Boca Raton, FL, 2004.
[14] R. Cont and E. Voltchkova, *A finite difference scheme for option pricing in jump diffusion and exponential Lévy models*, SIAM J. Numer. Anal. 43 (2005), pp. 1596–1626.
[15] C.W. Cryer, *The solution of a quadratic programming problem using systematic overrelaxation*, SIAM J. Control 9 (1971), pp. 385–392.
[16] M.A.H. Dempster and J.P. Hutton, *Fast numerical valuation of American, exotic and complex options*, Appl. Math. Finance 4 (1997), pp. 1–20.
[17] B. Eraker, M. Johannes, and N. Polson, *The impact of jumps in volatility and returns*, J. Finance 58 (2003), pp. 1269–1300.
[18] F. Fang and C.W. Oosterlee, *A novel pricing method for European options based on Fourier-cosine series expansions*, SIAM J. Sci. Comput. 31 (2008/2009), pp. 826–848.
[19] F. Fang and C.W. Oosterlee, *Pricing early-exercise and discrete barrier options by Fourier-cosine series expansions*, Numer. Math. 114 (2009), pp. 27–62.
[20] L. Feng and V. Linetsky, *Pricing options in jump-diffusion models: An extrapolation approach*, Oper. Res. 56 (2008), pp. 304–325.
[21] P.A. Forsyth and K.R. Vetzal, *Quadratic convergence for valuing American options using a penalty method*, SIAM J. Sci. Comput. 23 (2002), pp. 2095–2122.
[22] M.B. Giles and R. Carter, *Convergence analysis of Crank–Nicolson and Rannacher time-marching*, J. Comput. Finance 9 (2006), pp. 89–112.
[23] Y. d'Halluin, P.A. Forsyth, and G. Labahn, *A penalty method for American options with jump diffusion processes*, Numer. Math. 97 (2004), pp. 321–352.
[24] Y. d'Halluin, P.A. Forsyth, and K.R. Vetzal, *Robust numerical methods for contingent claims under jump diffusion processes*, IMA J. Numer. Anal. 25 (2005), pp. 87–112.
[25] A.D. Holmes and H. Yang, *A front-fixing finite element method for the valuation of American options*, SIAM J. Sci. Comput. 30 (2008), pp. 2158–2180.
[26] J. Huang and J.-S. Pang, *Option pricing and linear complementarity*, J. Comput. Finance 2 (1998), pp. 31–60.
[27] Y. Huang, P.A. Forsyth, and G. Labahn, *Methods for pricing American options under regime switching*, SIAM J. Sci. Comput. 33 (2011), pp. 2144–2168.
[28] S. Ikonen and J. Toivanen, *Operator splitting methods for American option pricing*, Appl. Math. Lett. 17 (2004), pp. 809–814.
[29] S. Ikonen and J. Toivanen, *Pricing American options using LU decomposition*, Appl. Math. Sci. (Ruse) 1 (2007), pp. 2529–2551.
[30] S. Ikonen and J. Toivanen, *Operator splitting methods for pricing American options under stochastic volatility*, Numer. Math. 113 (2009), pp. 299–324.
[31] K. Ito and K. Kunisch, *Parabolic variational inequalities: The Lagrange multiplier approach*, J. Math. Pures Appl. 85(9) (2006), pp. 415–449.

[32] K. Ito and J. Toivanen, *Lagrange multiplier approach with optimized finite difference stencils for pricing American options under stochastic volatility*, SIAM J. Sci. Comput. 31 (2009), pp. 2646–2664.
[33] P. Jaillet, D. Lamberton, and B. Lapeyre, *Variational inequalities and the pricing of American options*, Acta Appl. Math. 21 (1990), pp. 263–289.
[34] K.H. Karlsen and O. Wallin, *A semilinear equation for the American option in a general jump market*, Interfaces Free Bound. 11 (2009), pp. 475–501.
[35] S.G. Kou, *A jump-diffusion model for option pricing*, Manage. Sci. 48 (2002), pp. 1086–1101.
[36] Y. Kwon and Y. Lee, *A second-order finite difference method for option pricing under jump-diffusion models*, SIAM J. Numer. Anal. 49 (2011), pp. 2598–2617.
[37] Y. Kwon and Y. Lee, *A second-order tridiagonal method for American options under jump-diffusion models*, SIAM J. Sci. Comput. 33 (2011), pp. 1860–1872.
[38] R. Lord, F. Fang, F. Bervoets, and C.W. Oosterlee, *A fast and accurate FFT-based method for pricing early-exercise options under Lévy processes*, SIAM J. Sci. Comput. 30 (2008), pp. 1678–1705.
[39] A.-M. Matache, T. von Petersdorff, and C. Schwab, *Fast deterministic pricing of options on Lévy driven assets*, M2AN Math. Model. Numer. Anal. 38 (2004), pp. 37–71.
[40] A.-M. Matache, P.-A. Nitsche, and C. Schwab, *Wavelet Galerkin pricing of American options on Lévy driven assets*, Quant. Finance 5 (2005), pp. 403–424.
[41] A.-M. Matache, C. Schwab, and T.P. Wihler, *Fast numerical solution of parabolic integrodifferential equations with applications in finance*, SIAM J. Sci. Comput. 27 (2005), pp. 369–393.
[42] R.C. Merton, *Theory of rational option pricing*, Bell J. Econ. Manage. Sci. 4 (1973), pp. 141–183.
[43] R.C. Merton, *Option pricing when underlying stock returns are discontinuous*, J. Financ. Econ. 3 (1976), pp. 125–144.
[44] B.F. Nielsen, O. Skavhaug, and A. Tveito, *Penalty and front-fixing methods for the numerical solution of American option problems*, J. Comput. Finance 5 (2002), pp. 69–97.
[45] K.N. Pantazopoulos, E.N. Houstis, and S. Kortesis, *Front-tracking finite difference methods for the valuation of American options*, Comput. Econ. 12 (1998), pp. 255–273.
[46] H. Pham, *Optimal stopping, free boundary, and American option in a jump-diffusion model*, Appl. Math. Optim. 35 (1997), pp. 145–164.
[47] N. Rambeerich, D.Y. Tangman, A. Gopaul, and M. Bhuruth, *Exponential time integration for fast finite element solutions of some financial engineering problems*, J. Comput. Appl. Math. 224 (2009), pp. 668–678.
[48] R. Rannacher, *Finite element solution of diffusion problems with irregular data*, Numer. Math. 43 (1984), pp. 309–327.
[49] E.W. Sachs and A.K. Strauss, *Efficient solution of a partial integro-differential equation in finance*, Appl. Numer. Math. 58 (2008), pp. 1687–1703.
[50] S. Salmi and J. Toivanen, *An iterative method for pricing American options under jump-diffusion models*, Appl. Numer. Math. 61 (2011), pp. 821–831.
[51] D. Tavella and C. Randall, *Pricing Financial Instruments: The Finite Difference Method*, Wiley, New York, 2000.
[52] J. Toivanen, *Numerical valuation of European and American options under Kou's jump-diffusion model*, SIAM J. Sci. Comput. 30 (2008), pp. 1949–1970.
[53] J. Toivanen, *Finite difference methods for early exercise options*, in *Encyclopedia of Quantitative Finance*, R. Cont, ed., Wiley, New York, 2010.
[54] J. Toivanen, *A high-order front-tracking finite difference method for pricing American options under jump-diffusion models*, J. Comput. Finance 13 (2010), pp. 61–79.
[55] P. Wilmott, *Derivatives*, Wiley, Chichester, UK, 1998.
[56] L. Wu and Y.K. Kwok, *A front-fixing finite difference method for the valuation of American options*, J. Financ. Eng. 6 (1997), pp. 83–97.
[57] X.L. Zhang, *Numerical analysis of American option pricing in a jump-diffusion model*, Math. Oper. Res. 22 (1997), pp. 668–690.
[58] S.-P. Zhu, *An exact and explicit solution for the valuation of American put options*, Quant. Finance 6 (2006), pp. 229–242.
[59] W. Zhu and D.A. Kopriva, *A spectral element method to price European options. I. Single asset with and without jump diffusion*, J. Sci. Comput. 39 (2009), pp. 222–243.
[60] S.-P. Zhu and J. Zhang, *A new predictor–corrector scheme for valuing American puts*, Appl. Math. Comput. 217 (2011), pp. 4439–4452.
[61] R. Zvan, P.A. Forsyth, and K.R. Vetzal, *Penalty methods for American options with stochastic volatility*, J. Comput. Appl. Math. 91 (1998), pp. 199–218.