EUROPEAN OPTION VALUATION UNDER THE BATES PIDE IN FINANCE: A NUMERICAL IMPLEMENTATION OF THE GAUSSIAN SCHEME

Fazlollah Soleymani
Department of Mathematics, Institute for Advanced Studies in Basic Sciences (IASBS), Zanjan 45137–66731, Iran

Ali Akgül
Department of Mathematics, Art and Science Faculty Siirt University, Siirt, Turkey

Abstract. Models at which not only the asset price but also the volatility are assumed to be stochastic have received a remarkable attention in financial markets. The objective of the current research is to design a numerical method for solving the stochastic volatility (SV) jump–diffusion model of Bates, at which the presence of a nonlocal integral makes the coding of numerical schemes intensive. A numerical implementation is furnished by gathering several different techniques such as the radial basis function (RBF) generated finite difference (FD) approach, which keeps the sparsity of the FD methods but gives rise to the higher accuracy of the RBF meshless methods. Computational experiments are worked out to reveal the efficacy of the new procedure.

1. Introduction. Authors in [8] proposed a mathematical model in finance to calculate the option price of a European contract. After that, empirical evidences indicated that their model assumptions on the log–normality of the return of the underlying asset and constant volatility are usually inconsistent with market prices, for more refer to the discussions in [26].

Basically in such models, the financial problem is formulated as a partial differential equation (PDE) of second order with nonconstant coefficients. On the other hand, formulation of models based on a (multidimensional) partial integro–differential equation (PIDE) is naturally more fruitful since they do not only cover the asset price and volatility to be stochastic but also take the jumps into consideration, which directly affects to pricing as reliably as possible, [25, 29].

For the valuation of options with long maturities, stochastic volatility (SV) models (based upon pure Brownian motion modeling) like the Heston stochastic volatility model [20] are a common means to introduce such variability. For short maturities, the Heston model often require excessively large volatilities to explain the market prices of options. A modern way to resolve this, is to incorporate jumps in the asset price model, like the classical Merton jump–diffusion model [30].

The Bates model [6] combines the Merton and the Heston models. This asset price model includes both jumps and stochastic volatility and it is therefore popular

2010 Mathematics Subject Classification. Primary: 91B25; Secondary: 65M22.

Key words and phrases. Jump–diffusion model, nonlocal integral, method of lines, Mathematica, Gaussian function.

* Corresponding author.
for valuing options with short as well as long maturities. This model is also refereed to Bates–Scott model, since it was originally proposed independently at the same time by Bates in [6] and Scott in [37].

Although the solution of such models for specific options and under mild assumptions can be found analytically, in general the pricing procedure is done via computational schemes, for instance refer to [17] and the references therein.

There are a number of papers for this task. Authors in [4] investigated a finite element (FE) approach for tackling this problem while standard FD method and its adaptive formulation discussed in [44]. Authors in [36] also investigated an implicit–explicit computational scheme for solving the Bates model using an operator splitting for the American–style options. For some background, refer to [2].

Two asset contingent claims based on jump diffusion were computed in [11] by applying a Markov chain technique that could be basically considered as an explicit FD scheme. In that technique, the presence of a not–sparse system of linear equations in discretizing of the associated integral operator of the PIDE is prevented by mixing a fixed–point method and a Fast Fourier Transform (FFT). However, the approach is computationally intensive.

Lately, authors in [14] proposed a transformation on this model to remove the presence of the mixed derivative term and make the model with constant coefficients at the cost of transfiguring the standard rectangular–like numerical domain into a rhomboid–like domain. Furthermore, a FD method with uniform grid of nodes possessing quadratic convergence as well as a time–stepping method required a refined step–size (to control the stability) have been suggested. For more, an interested reader may refer to [3,12].

Radial basis function (RBF) meshless schemes are an important portion of an ever–growing field of numerical analysis. First discussed by Hardy (see e.g. [19]), in 1968, such schemes permit the scattered data to straightforwardly be applied in calculations. This matter was pursued before by applying polynomial interpolation, but the RBF interpolation was discussed to work in several cases, at which the polynomial interpolation has failed, [15, chapters 1-2]. Recalling that the RBF meshfree schemes are divided into two main branches, i.e., global and local methods.

Once higher accuracy orders are adequate in a typical problem, or if the solutions are needed to have finite–smoothness only, RBF–FD approximations are an attractive choice in contrast to the global RBF meshfree methods since they provide better results with respect to accuracy and burdensome [16]. These approximations are produced from RBF interpolation on local collections of points such that the obtained derivative matrices are sparse exactly similar to the standard FD scheme, but with better convergence properties, [41].

Against the conventional procedures, the RBF–FD scheme could basically tackle irregular geometries and scattered node layouts. Furthermore, their locality provides more flexibility with respect to local refinement techniques such as the global RBF schemes.

In this paper, solving the well–known Bates model is tackled numerically. The underlying problem has a discontinuity in the initial condition and is degenerate in one of the sides of the domain, i.e., when the variance become zero. Accordingly, it is necessary to design a new efficient method to be fast and robust.

To do this, the weights of the RBF–FD scheme using the Gaussian RBF (GRBF) for unstructured points are written and applied for the 1st and 2nd derivatives of a
function. It is well-known that the GRBF is defined by [7]:
\[ \phi(r_i) = e^{-(r_i)^2}, \quad i = 1, 2, \ldots, m, \]  
where \( r_i = \|\mathbf{x} - \mathbf{x}_i\|_2 \) is the Euclidean distance and \( c \) is the shape parameter. This parameter is the key element in the accuracy of the estimations, see a study about this parameter in [39].

The motivation of this paper is to solve the time-dependent PIDE arising from the Bates model including a nonlocal integral. We apply the weights of the GRBF–FD method for spatially discretizing the PIDE problem. Moreover, nonuniform meshes are applied along the spatial variables as well as a Krylov subspace method is used to tackle the obtained system of ODEs as accurately and efficiently as possible.

The computation of matrix exponential function on a vector, with a Krylov subspace scheme fundamentally includes two parts. The first part calculates a suitable Krylov subspace and the second calculates the matrix exponential of a smaller matrix using typical schemes.

After this introduction regarding the importance of the model in finance and furnishing the structure of the proposed procedure, the remaining sections are organized as comes next. Section 2 gives the mathematical formulation for the Bates model as well as its initial/boundary conditions and the distribution of the nodes. It also provides the weights of the GRBF–FD scheme in the presence of a stencil containing three points.

The discretization of the PIDE along the spatial variable using the RBF–FD methodology is brought forward in Section 3. A system of (stiff) ODEs is constructed which is of large size, which is solved via an application of the matrix exponential function on a vector in Section 3 as well. In Section 4, we compute European options in several computational tests, that establish a good efficacy of the presented approach. Lastly, Section 5 ends the paper. An important part of the paper is the Appendix, at which we provide a simple yet effective Mathematica program for implementing our scheme.

2. Bates model, nodes’ distribution and weights. Let us consider jumps in the volatility and investigate the Bates model by considering that the value of asset \( S_t \) and its variance \( V_t \) read the coupled stochastic differential equations (SDEs) below [6]:
\[ dS_t = (r - q - \lambda \zeta)S_t dt + \sqrt{V_t}S_t dW^1_t + (\eta - 1)S_t dN_t, \]
\[ dV_t = \kappa(\theta - V_t)dt + \sigma \sqrt{V_t}dW^2_t, \]
wherein \( W^1_t \) and \( W^2_t \) are two standard Brownian motions with \( dW^1_t dW^2_t = \rho dt \). Here \( N_t \) is the Poisson process. Furthermore, \( r \) is the risk-free interest rate, \( q \) is the dividend yield, \( \zeta \) is the mean jump, \( \eta \) is the jump size, i.e., a random variable measuring the jump amplitude and is assumed to be log–normally distributed in \( (0, +\infty) \), \( \lambda \) is the intensity of the Poisson process, \( \kappa \) is the rate of reversion of the variance \( V_t \), while \( \theta \) and \( \sigma \) are the mean level and (initial) volatility constant, respectively.

Following a usual derivation based on Itô’s lemma and the no arbitrage requirement, we obtain the following PIDE for the price \( u(s, v, \tau) \) of a European option:
\[ \frac{\partial u(s, v, \tau)}{\partial \tau} = \frac{1}{2} vs \frac{\partial^2 u(s, v, \tau)}{\partial s^2} + \frac{1}{2} \sigma^2 v \frac{\partial^2 u(s, v, \tau)}{\partial v^2}. \]
\[\begin{align*}
&+ \rho \sigma v s \frac{\partial^2 u(s, v, \tau)}{\partial s \partial v} \\
&+ (r - q - \lambda \zeta) s \frac{\partial u(s, v, \tau)}{\partial s} + \kappa (\theta - v) \frac{\partial u(s, v, \tau)}{\partial v} \\
&- (r + \lambda) u(s, v, \tau) + \lambda \int_0^\infty u(s\eta, v, \tau) g(\eta) d\eta \\
&:= \mathcal{L} u(s, v, \tau),
\end{align*}\]  

where \(\tau = T - t\) and \(\mathcal{L} u(s, v, \tau) = \mathcal{L}_D u(s, v, \tau) + \lambda \mathcal{L}_I u(s, v, \tau)\), and \(\mathcal{L}_I u(s, v, \tau) = \int_0^{+\infty} u(s\eta, v, \tau) g(\eta) d\eta\). The 1D probability density function (PDF) \(g(\eta)\) is defined by

\[g(\eta) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(\ln(\eta) - \gamma)^2}{2\sigma^2}}.\]

Here the mean \(\gamma\) and the standard deviation \(\sigma\) are positive constants. Furthermore, we have \(\zeta = e^{(\gamma + \frac{1}{2}\sigma^2)} - 1\).

The PIDE problem (3) is well posed as a Cauchy initial value problem. The initial conditions for the vanilla put and call options are defined by

\[u(s, v, 0) = p(s) = \max\{E - s, 0\}, \quad u(s, v, 0) = p(s) = \max\{s - E, 0\},\]

respectively, wherein \(E\) denotes the price of strike. In fact, the function \(p\) specifies the value of the option at expiry.

Conditions on boundaries (sides) for (3) are important since the aim is to find a (viscosity) solution of the pricing PIDE (3), which also reads the given side conditions, see e.g. [31, Chapter 1.2]. For a call option, we have [5]:

\[\begin{align*}
u(s, v, \tau) &\simeq 0, \quad s \to 0, \\
u(s, v, \tau) &\simeq s_{\text{max}} e^{(-q\tau)} - E e^{(-r\tau)}, \quad s \to +\infty, \\
\frac{\partial u(s, v, \tau)}{\partial v} &\simeq 0, \quad v \to +\infty.
\end{align*}\]

In case of a European put option, the boundary conditions are defined by:

\[\begin{align*}
u(s, v, \tau) &\simeq E e^{(-r\tau)} - s_{\text{max}} e^{(-q\tau)}, \quad s \to 0, \\
u(s, v, \tau) &\simeq 0, \quad s \to +\infty, \\
\frac{\partial u(s, v, \tau)}{\partial v} &\simeq 0, \quad v \to +\infty.
\end{align*}\]

In the Bates model, the conditions for boundaries are provided for \(s \to 0, s \to +\infty\) and \(v \to +\infty\). As a matter of fact, for \(v = 0\) the PIDE (3) is degenerate and therefore it is not yet that obvious for which condition the function \(u(s, v, \tau)\) has to read. Accordingly, no condition on this boundary is imposed, [5].

**Theorem 2.1.** [9] Consider the Lipschitz continuity on (3). Afterwards, for every initial value \(p \in C((0, +\infty)^N) \cap W^{1,\infty}_\text{pol}((0, +\infty)^N)\), the PIDE (3) has a unique solution of viscosity \(u\), which is a member of \(L^\infty(0, T; W^{1,\infty}_\text{pol}((0, +\infty)^N)\). Here, the index pol indicates a polynomial growth of the considered norms at infinity, while \(N\) denotes the truncated domain inside the main domain.

**Corollary 1.** [9] (Positivity Condition) The principle of comparison holds and we obtain \(u(s, v, \tau) \geq 0\) whenever \(p(s) \geq 0\).
For this option pricing problem, the spatial domain should be written as: \( \Omega = [0, s_{\text{max}}] \times [0, v_{\text{max}}] \), where \( s_{\text{max}} \) and \( v_{\text{max}} \) are positive real constants considered large enough so as to minimize the error of the main domain truncation.

Since (3) is degenerate at \( v = 0 \), its initial condition is not differentiable at the price of strike and the numerical domain is a large rectangle, it is requisite to consider non–uniform grids, at which the nodes’ distribution for the application of method of lines (MOL) are non–equidistant. This clearly helps in producing results of higher accuracy with adapting to the hotzone of the problem. Some discussion on the importance of such meshing are given in [4, 23]. Among many ways to produce grids with a concentration on a special zone, we here adopt the well–resulted non–uniform smooth grid of points proposed in [23] for the two spatial variables \( s \) and \( v \).

Let \( \{s_i\}_{i=1}^m \) be a partition for \( s \in [s_{\text{min}}, s_{\text{max}}] \). Then, the meshing along this variable can be defined as follows:

\[
s_i = \Psi(\xi), \quad 1 \leq i \leq m,
\]

wherein \( \xi_{\text{min}} = \xi_1 < \xi_2 < \cdots < \xi_m = \xi_{\text{max}} \) are \( m \) uniform nodes with the following specifications:

\[
\xi_{\text{min}} = \sinh^{-1}\left(\frac{s_{\text{min}} - s_{\text{left}}}{d_1}\right), \quad \xi_{\text{int}} = \frac{s_{\text{right}} - s_{\text{left}}}{d_1} - d_1 \sinh^{-1}\left(\frac{s_{\text{max}} - s_{\text{right}}}{d_1}\right). \quad \xi_{\text{max}} = \xi_{\text{int}} + \sinh^{-1}\left(\frac{s_{\text{max}} - s_{\text{right}}}{d_1}\right).
\]

According to \( \Omega \), \( s_{\text{min}} = 0 \). The parameter \( d_1 > 0 \) is a constant that conducts the fraction of the mesh nodes \( s_i \) that are situated around the price of strike. Here the zoom function is furnished by:

\[
\Psi(\xi) = \begin{cases} 
  s_{\text{left}} + d_1 \sinh(\xi), & \xi_{\text{min}} \leq \xi < 0, \\
  s_{\text{left}} + d_1 \xi, & 0 \leq \xi \leq \xi_{\text{int}}, \\
  s_{\text{right}} + d_1 \sinh(\xi - \xi_{\text{int}}), & \xi_{\text{int}} < \xi \leq \xi_{\text{max}}.
\end{cases}
\]

In this work, we used the value \( d_1 = \frac{E}{15} \), while \( s_{\text{left}} = \max\{0.5, e^{-0.25T}\} \times E, \ s_{\text{right}} = E, \ [s_{\text{left}}, s_{\text{right}}] \subset [0, s_{\text{max}}] \), and \( s_{\text{max}} = 3E \).

If \( \{v_j\}_{j=1}^n \) stands for a partition along the spatial variable \( v \), then the meshing in this direction which should be focused around zero is expressed by:

\[
v_j = d_2 \sinh(c_j), \quad 1 \leq j \leq n,
\]

wherein \( d_2 > 0 \) is a constant that handles the fraction of the mesh centers/nodes \( v_j \) that locate near \( v = 0 \). Here, we selected \( d_2 = \frac{v_{\text{max}}}{150} \), where \( v_{\text{max}} = 1 \).

Herein \( c_j \) are equidistant points defined as \( c_j = (j-1)\Delta \varsigma, \Delta \varsigma = \frac{1}{n-1} \sinh^{-1}\left(\frac{v_{\text{max}}}{d_2}\right) \), for any \( 1 \leq j \leq n \).

We state that a detailed study into good choices for parameters \( d_1 \) and \( d_2 \) may be interesting, but this is beyond the scope of the current research.

The final grid with a clear concentration around the hotzone is attained using a Cartesian product of the grids along \( s \) by (7), and \( v \) by (9) as comes next:

\[
\text{Discretized } \Omega = \{s_i\} \times \{v_j\}, \quad 1 \leq i \leq m, \quad 1 \leq j \leq n.
\]

To calculate the weights of the GRBF–FD method for the 1st derivative, let us consider an unstructured stencil having three nodes. In order to derive the weighting coefficients \( \alpha_i \) for the 1st derivative, we may include three non–equispaced nodes as \( \{x_i - h, x_i, x_i + \omega h\}, (\omega > 0, h > 0) \), and write

\[
 u'(x_i) \approx \alpha_{i-1}u(x_{i-1}) + \alpha_iu(x_i) + \alpha_{i+1}u(x_{i+1}) = \hat{u}'(x_i). \quad (11)
\]

Substituting the function \( u \) by the RBFs, centered at \( x_{i-1} = x_i - h, x_i, \) and \( x_{i+1} = x_i + \omega h \), yields a linear system of three equations. Hence, the weights of the
GRBF–FD formula for the 1st derivative once \( c \gg h \) would be
\[
\alpha_{i-1} = \frac{\omega (h^2 (2\omega - 5) - 3c^2)}{3c^2 h (\omega + 1)}, \quad \alpha_i = \frac{\omega - 1}{h\omega} - \frac{2h(\omega - 1)}{3c^2}, \quad \alpha_{i+1} = \frac{h^2 (5\omega - 2) + \frac{3}{\omega}}{3h(\omega + 1)}.
\]
To find the weighting coefficients for the 2nd derivative of a function, we may similarly write
\[
u''(x_i) \simeq \beta_{i-1} u(x_{i-1}) + \beta_i u(x_i) + \beta_{i+1} u(x_{i+1}) = \hat{u}''(x_i).
\]
Now by constructing a set of three equations and solving in the limit \( c \gg h \), we obtain:
\[
\beta_{i-1} = \frac{2(2\omega - 2\omega^2 + 5)}{3(\omega + 1)} \left( \frac{2c^2}{\omega} \right),
\beta_i = \frac{2(-2\omega^2 + \omega - 2)}{3\omega} \left( \frac{3c^2}{\omega} \right),
\beta_{i+1} = \frac{6c^2 + 2h^2 (5\omega - 4) + 2}{3c^2 h^2 \omega (\omega + 1)}.
\]
The convergence speed for the 1st derivative is at least two, while for the 2nd derivative, it is of first convergence speed. The point is, these orders have been obtained on non–equidistant grids and because of that not much higher rates can be attained by employing three nodes at the same time.

The GRBF–FD weights obtained here are useful to apply MOL in Section 3.

3. Method of solution. Under the Bates model (3), a 2D parabolic PIDE was given in (3) for the values of European–style options with the spatial variables representing the underlying asset price and its instantaneous variance. Noting that in (3), the differential operator is of the convection–diffusion type.

Discretization is the process of transferring the continuous Bates model (3) into discrete counterparts. Whenever continuous data is discretized, there is always some amount of discretization error. The aim is to decrease the amount to a level assumed negligible for the existing modeling aims [31]. This is why the GRBF–FD method with non–uniform nodes’ distribution is used here.

By considering the notion of derivative matrices, we first form the following matrices of differentiation [38] along \( s \):
\[
D_s = \begin{pmatrix}
\alpha_{1,1} & \alpha_{1,2} & 0 & 0 & 0 \\
\alpha_{2,1} & \alpha_{2,2} & \alpha_{2,3} & 0 & 0 \\
\ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & \alpha_{m-1,m-2} & \alpha_{m-1,m-1} & \alpha_{m-1,m} \\
0 & 0 & 0 & \alpha_{m,m-1} & \alpha_{m,m}
\end{pmatrix}_{m \times m},
\]
and
\[
D_{ss} = \begin{pmatrix}
\beta_{1,1} & \beta_{1,2} & 0 & 0 & 0 \\
\beta_{2,1} & \beta_{2,2} & \beta_{2,3} & 0 & 0 \\
\ddots & \ddots & \ddots & \ddots & \ddots \\
0 & 0 & \beta_{m-1,m-2} & \beta_{m-1,m-1} & \beta_{m-1,m} \\
0 & 0 & 0 & \beta_{m,m-1} & \beta_{m,m}
\end{pmatrix}_{m \times m},
\]
wherein \( \alpha_{i,j} \) and \( \beta_{i,j} \) are the weights computed throughout mesh points via (12) and (14), respectively.

The derivative matrices containing the GRBF–FD weights of the 1st and 2nd derivatives along the other spatial variable, i.e., the variance \( v \) could be written similarly as (15)–(16). Noting that for the two nodes located on the boundaries,
we only apply the RBF–FD weights based on the two nearby points to keep the matrices tri–diagonal as well. This affects on $\alpha_{1,1}$, $\alpha_{1,2}$, $\alpha_{m,m-1}$, $\alpha_{m,m}$, $\beta_{1,1}$, $\beta_{1,2}$ $\beta_{m,m-1}$, and $\beta_{m,m}$. A detailed implementation of this part is illustrated thoroughly in a Mathematica code provided in the Appendix.

It is known that the multivariate functions are related to the multivariate arrays, or tensors, [18]. In fact, tensor is a $d$–dimensional array, i.e., a polynlinear form, which forms a linear vector space. Hence, to construct our GRBF–FD method for our 2D problem more elegantly, we may rely on the Kronecker product (denoted by $\otimes$) of the differentiation matrices, see for more [24].

As an illustration, the differentiation matrix corresponding to the mixed derivative term $\frac{\partial^2}{\partial s \partial v}$ can be obtained as follows:

$$
(D_{sv})_{N \times N} = (D_s)_{m \times m} \otimes (D_v)_{n \times n},
$$

where $N = m \times n$. Other derivative terms in (3) can be calculated in a similar manner.

Summarizing all the weights coming from different terms together give rise to a coefficient matrix $\Upsilon$. Therefore, the PIDE model (3) without the integral part can be represented as a set of (linear) homogeneous ODEs as follows:

$$
\dot{U}(\tau) = \Upsilon U(\tau), \quad 0 < \tau \leq T.
$$

(18)

The system matrix $\Upsilon$ in (18) is not time–varying but it has not yet included the effect of jumps (due to the discretization of the internal operator) and the boundary conditions. These would be discussed later in the section.

3.1. The nonlocal integral and side conditions. The integral operator, which stems from the jumps, couples all option values in the asset price direction.

To discretize this part, by a linear interpolation for $u(s\eta, v, \tau)$ among the non–uniform computational grid points, the following expression from (3) is discretized

$$
\mathcal{L}_I(u) = \int_0^\infty u(s\eta, v, \tau) g(\eta) d\eta.
$$

(19)

A change of variable $z = s\eta$, is now taken into consideration to transfigure (19) into the following equivalent form:

$$
\mathcal{L}_I(u) = \int_0^\infty u(z, v, \tau) g \left( \frac{z}{\sigma} \right) \left( \frac{1}{\sigma} \right) dz.
$$

(20)

Using linear interpolation, we obtain an approximation of (20) as follows:

$$
\mathcal{L}_I(u) \approx \sum_{i=1}^{m-1} M_{i,l},
$$

(21)

at each grid point $s_i$, $i = 2, 3, \ldots, m - 1$, where

$$
M_{i,l} = \int_{s_l}^{s_{l+1}} \left( \frac{s_{l+1} - z}{\Delta s_l} u(s_l, v, \tau) + \frac{z - s_l}{\Delta s_l} u(s_{l+1}, v, \tau) \right) g \left( \frac{z}{\sigma} \right) \left( \frac{1}{\sigma} \right) dz,
$$

(22)

wherein $\Delta s_l = s_{l+1} - s_l$ is the (non–uniform) step size. Taking into consideration the log–normal distribution $g(\eta)$ based on (4), it is possible to write down:

$$
M_{i,l} = \frac{1}{\sqrt{2\pi\sigma}} \int_{s_l}^{s_{l+1}} \left( \frac{s_{l+1} - z}{\Delta s_l} u(s_l, v, \tau) + \frac{z - s_l}{\Delta s_l} u(s_{l+1}, v, \tau) \right)
$$
Finding the solution of (23) symbolically [33] yields:

\[
M_{i,l} = \frac{1}{2\Delta s_l} \left( e^{\left(\gamma + \frac{s_i^2}{2}\right)} \left( -\text{erf}\left( \frac{-\ln\left(\frac{s_l + s_i}{s_l}\right) + \gamma + \hat{\sigma}^2}{\sqrt{2\hat{\sigma}}} \right) \right) + \text{erf}\left( \frac{-\ln\left(\frac{s_l + s_i}{s_l}\right) + \gamma + \hat{\sigma}^2}{\sqrt{2\hat{\sigma}}} \right) \right) \left( s_l u(s_l, v, \tau) - s_l u(s_l + 1, v, \tau) \right)
\]

\[
\times e^{-\left(\frac{\ln\left(\frac{s_i}{2s^2}\right)}{2s^2}\right)/s}dz.
\]

(23)

wherein the error function \(\text{erf}(\cdot)\) is the integral of the Gaussian distribution and is defined by:

\[
\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt.
\]

(25)

By considering the discretization of the integral part into the system (18), we get

\[
\dot{U}(\tau) = \tilde{\Upsilon}U(\tau), \quad 0 < \tau \leq T.
\]

(26)

Now, this is the time to incorporate the boundaries. Side conditions are important in such models, see [28] to observe their importance. In this paper, we proceed with an approach at which two conditions at two sides of the domain are incorporated while for the other two sides, the discretized equations (with the discretized integral term) are considered.

In fact, the nodes which are located on the boundary \(v = 0\) are considered as interior nodes and we take a fact into consideration that they must read the PIDE model (with the integral term). That is to say, no boundary condition is imposed separately, and we incorporate the semi–discretized equations at this boundary as the true boundary, see a similar strategy in [13].

In this work in order to reduce the computational burdensome, we apply a similar strategy as in [12] and impose the boundaries only for two sides of the computational domain, i.e., once \(s = 0\) and \(s = s_{\text{max}}\), by considering their differentiation forms along the temporal variable to be zero. As in the case \(v = 0\), once \(v = v_{\text{max}}\), no side condition is prescribed, and the discretized equation is assumed as a boundary. See Appendix for a detailed implementation for interested readers.

Collecting the semi–discrete approximations and boundary conditions to the grid point values, gives rise to a set of linear ODEs of the following form

\[
\dot{U}(\tau) = \tilde{\Upsilon}U(\tau), \quad 0 < \tau \leq T.
\]

(27)

The ODE system (27) is complemented with an initial vector \(U(0)\) that is given by the values of the payoff function \(p\) at the spatial grid points.

The system (27) is locally well–posed, i.e., there exists a unique solution (depending on the initial condition), which read (27). As a matter of fact, we can show that \(G(\tau, U) = \tilde{\Upsilon}U(\tau)\) is continuous in its first argument \(\tau\), and locally uniformly Lipschitz continuous in its second argument. To be more precise, we could write

\[
\|G(\tau, U_1) - G(\tau, U_2)\| \leq \|\tilde{\Upsilon}\|\|U_1 - U_2\|.
\]

(28)
and therefore the Lipschitz constant would be $L = \|\tilde{\Upsilon}\|$ for a spectral matrix norm, which is the largest singular value of $\tilde{\Upsilon}$.

3.2. A Krylov subspace method. It is famous that the time-stepping methods such as the Runge-Kutta and the generalized linear methods (GLMs) are sensitive to the choice of the step size or in implicit forms require high computational burden due to solving system of nonlinear algebraic equations (NAE) of large sizes.

When the time interval is large, many time steps may be required in order to obtain a given accuracy, which is so time consuming. In lieu of applying such methods for system of ODEs, the authors in [35, 40] investigated the exponential time integration (ETI) method to solve PIDEs arising from both the Black–Scholes and Merton’s models. The merit of the ETI scheme is that it is a one step method and hence we need not to discuss its stability and temporal discretized accuracy.

The closed-form solution of the set of stiff ODEs (27) can be represented as [27]:

$$U(\tau) = e^{\tau \tilde{\Upsilon}} U(0).$$

(29)

A wide variety of techniques for calculating the exponential of a matrix could be observed in work [32]. Two of the main ones are the limit formulation [43]:

$$e^{\tilde{\Upsilon}} = \lim_{s \to \infty} (I + A/s)^s,$$

(30)

or the scaling and squaring formulation

$$e^{\tilde{\Upsilon}} = \left(e^{A/2^s}\right)^{2^s},$$

(31)

which include higher burdensome for big-sized system matrix $\tilde{\Upsilon}$.

Remarking that when $\tau \neq 1$, by using a transformation $\tilde{\Upsilon} \leftarrow \tau \tilde{\Upsilon}$, we can easily have the same formulation as in (30) or (31).

In [35, 40], the corresponding matrix exponential function was directly computed by the scaling and squaring algorithm with Padé’s approximation [21] which has an $O(N^3)$ complexity, where $N$ is the matrix size. It is noticed that what we ultimately require is the multiplication of a matrix exponential and a vector, but not the exact matrix exponential. For such case, the recently used Krylov subspace methods such as the one in [34] work well.

Let us repeat that another clear advantage of such an approach is that it is a one-step method (step-size-free method), i.e., the final solution regardless the value of the $T$ (the final value for the temporal value) can be found immediately. This heavily reduces the effort in contrast to the methods in the class of GLM, which needs more number of temporal nodes in such a case.

Krylov subspace techniques are most improved in both theory and practice. Thus, we can use a Krylov method at which no matrix products or divisions but matrix-vector multiplications are needed in the whole solving procedure. The main concept is to roughly project the exponential of a matrix onto a small Krylov subspace by applying the well-known Arnoldi process, see for more [10, 21].

Hopefully, this method has already been coded efficiently in the programming package Mathematica [1] and can be called on using the following command as we do in this paper:

MatrixExp[systemmatrix, payoff, Method -> "Krylov"];

The real part of the largest eigenvalue of $\tilde{\Upsilon}$ (in the absolute value sense) plays a central rule in stiffness of the system as well as the ill-conditioning of the system matrix. In fact, whatever real part of the largest eigenvalue is larger, the application
of all solvers, time–stepping with refined temporal step–size or the step–size–free
methods, is harder.

| Scheme  | m  | n  | N  | k  | u   | ε         | Time (s) |
|---------|----|----|----|----|-----|-----------|----------|
| SFD–EM  | 20 | 20 | 400| 400| 8.7001 | 1.947 × 10⁻¹ | 0.91     |
|         | 40 | 25 | 1000| 2000| 8.5974 | 2.973 × 10⁻¹ | 2.67     |
|         | 40 | 40 | 1600| 2000| 8.6739 | 2.209 × 10⁻¹ | 5.39     |
|         | 65 | 45 | 2925| 4000| 8.8609 | 3.397 × 10⁻² | 15.77    |
|         | 80 | 50 | 4000| 10000| 8.8745 | 2.036 × 10⁻² | 33.95    |
| HFM–DM  | 10 | 10 | 100 | 250| 8.3465 | 5.483 × 10⁻¹ | 0.45     |
|         | 15 | 15 | 225 | 250| 8.6980 | 1.968 × 10⁻¹ | 0.64     |
|         | 25 | 20 | 500 | 400| 8.8601 | 3.473 × 10⁻² | 1.12     |
|         | 30 | 30 | 900 | 600| 8.8705 | 2.431 × 10⁻² | 2.03     |
|         | 50 | 30 | 1500| 2000| 8.8852 | 9.624 × 10⁻³ | 5.41     |
|         | 80 | 30 | 2400| 5000| 8.8905 | 4.320 × 10⁻³ | 13.10    |
| GRBF–FDI| 10 | 10 | 100 | NR | 8.4182 | 4.766 × 10⁻¹ | 0.08     |
|         | 15 | 15 | 225 | NR | 8.7578 | 1.370 × 10⁻¹ | 0.13     |
|         | 25 | 20 | 500 | NR | 8.8513 | 4.355 × 10⁻² | 0.14     |
|         | 30 | 30 | 900 | NR | 8.8659 | 2.891 × 10⁻² | 0.18     |
|         | 60 | 30 | 900 | NR | 8.8905 | 4.321 × 10⁻³ | 0.58     |
|         | 80 | 30 | 2400| NR | 8.8952 | 3.305 × 10⁻⁴ | 1.39     |

Table 1. Numerical reports of call vanilla option pricing for Problem 4.1.

4. Numerical results. In this section, we concern the comparison of the existing
frequently used schemes in estimating the Bates model (3), namely:

• The standard FD method with equidistant grids and second–order spatial
convergence and the Euler’s time–stepping method (denoted by SFD–EM) in [12].

• The non–uniform FD method (denoted by HFM–DM) based on the non–
equidistant grids in [22] for the Heston case. Here the inputs discussed in
Section 2 are used along with the Douglas method (i.e., its free parameter is
zero).

• The scalable algebraic multigrid method (denoted by AMG–STS) from [36],

• And the new procedure described in this work (denoted by GRBF–FDI).

In this section, all the compared methods (except AMG–STS) have been coded
in a same computational domain and a programming language.

The shape parameter for the GRBF–FDI method is selected by:

\[ c_{\text{for } s} = 3 \max \{ \Delta s_i \}, \quad c_{\text{for } v} = 6 \max \{ \Delta v_j \}, \]

where \( \Delta s_i \) and \( \Delta v_j \) are the increment along \( s \) and \( v \), respectively.

All written and applied snippets are implemented in Mathematica 11.0 and
therefore easily comparable in terms of efficacy, [42]. Additionally, if the position
\( (s, v) \) is not a point of the mesh, the estimated solution at this node is computed
by a built–in interpolation command `Interpolation[]` in the programming package Mathematica 11.0.

The following remarks are in order:

• We performed the experiments on an office laptop having Windows 7 Ultimate equipped Intel(R) Core(TM) i5–2430M CPU 2.40GHz processor and 16.00 GB of RAM on a 64–bit operating system.
• In the following tables “NR” stands for “not required”, viz, our solver applied to tackle the semi–discretized set of ODEs is independent of the choice of the step size. Additionally, $k + 1$ is the total number of nodes for temporal discretization.
• $Re(\lambda_{\text{max}})$ indicates the real part of the largest eigenvalue (in absolute value sense) of the system matrix $\tilde{\Upsilon}$ in absolute value sense.

The absolute error is reported via

$$
\varepsilon = |u_{\text{approx}}(s, v, \tau) - u_{\text{ref}}(s, v, \tau)|,
$$

wherein $u_{\text{ref}}$ and $u_{\text{approx}}$ are the exact and estimated solutions, respectively. Moreover, the root mean square relative error (RMSRE) of $\mathcal{N}$ calculated solutions $u_{\text{approx}}$
whose expected values are $u_{\text{ref}}$, respectively, is reported by
\[
\epsilon = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left( \frac{1}{N} \left| u_{\text{approx}}(s_i, v, \tau) - u_{\text{ref}}(s_i, v, \tau) \right|^2 \right)}.
\]
(34)

**Problem 4.1.** [22] Numerical methods are compared for a European call option having the following settings

\[
T = 1, \ E = 100, \ r = 0.025, \ q = 0, \ \theta = 0.04, \ \kappa = 1.5, \ \sigma = 0.3, \ \lambda = 0, \ \rho = -0.9.
\]
(35)

For the call scenario, we compare the price obtained through each scheme with the price obtained through the very refined implementation of FFT approach, i.e., 8.894869, which we took as the reference in the hotzone $(s, v) = (100, 0.04)$.

The results for this test are put together in Table 1. The option price could be calculated with a high level of precision in a short piece of time using GRBF–FDI, while the other solvers need more and more spatial/temporal nodes to achieve a good numerical result.

It is requisite to plot the numerical solution in order to reveal its smoothness and positivity throughout the whole numerical domain. To do this, once $m = 80$ and $n = 30$, the computational results are plotted in Figure 1.

**Problem 4.2.** Here we re–compare the methods using the same settings as in Problem 4.1 but for a put option. The reference values at $s = \{80, 90, 100, 110, 120\}$ and $v = 0.04$ are roughly 17.9638, 10.7886, 6.4247, 3.8945, and 2.4193, respectively.

| Scheme   | $m$ | $n$ | $N$   | $k$   | $Re(\lambda_{\text{max}})$ | $\epsilon$ | Time (s) |
|----------|-----|-----|-------|-------|-----------------------------|------------|----------|
| SFD–EM   | 20  | 20  | 400   | 400   | -516.30                     | $2.582 \times 10^{-2}$ | 0.15     |
|          | 40  | 25  | 1000  | 2000  | -2438.80                    | $2.368 \times 10^{-2}$ | 0.99     |
|          | 40  | 40  | 1000  | 2000  | -2512.03                    | $2.109 \times 10^{-2}$ | 1.55     |
|          | 65  | 45  | 2925  | 4000  | -7041.97                    | $4.009 \times 10^{-3}$ | 5.01     |
|          | 80  | 50  | 4000  | 10000 | -10932.60                   | $2.972 \times 10^{-3}$ | 19.43    |
| HFM–DM   | 10  | 10  | 100   | 250   | -109.47                     | $5.948 \times 10^{-2}$ | 0.10     |
|          | 15  | 15  | 225   | 250   | -298.15                     | $2.450 \times 10^{-2}$ | 0.12     |
|          | 25  | 20  | 500   | 500   | -951.58                     | $3.982 \times 10^{-3}$ | 0.22     |
|          | 30  | 30  | 900   | 1000  | -1418.65                    | $3.015 \times 10^{-3}$ | 0.51     |
|          | 80  | 30  | 2400  | 10000 | -11134.30                   | $4.722 \times 10^{-4}$ | 11.12    |
| GRBF–FDI | 10  | 10  | 100   | NR    | -238.16                     | $8.173 \times 10^{-2}$ | 0.09     |
|          | 15  | 15  | 225   | NR    | -639.62                     | $3.579 \times 10^{-2}$ | 0.10     |
|          | 25  | 20  | 500   | NR    | -2032.56                    | $1.210 \times 10^{-2}$ | 0.15     |
|          | 30  | 30  | 900   | NR    | -3030.1                     | $8.521 \times 10^{-3}$ | 0.19     |
|          | 60  | 30  | 1800  | NR    | -1321.14                    | $1.746 \times 10^{-3}$ | 0.57     |
|          | 80  | 30  | 2400  | NR    | -24023.6                    | $5.879 \times 10^{-4}$ | 1.28     |

Table 2. Numerical reports of put vanilla option pricing for Problem 4.2.
Table 3. Parameter settings for the Bates model.

| Descriptions                              | Parameters | Values |
|-------------------------------------------|------------|--------|
| Correlation between the Brownian motions  | ρ          | -0.5   |
| Rate of interest                          | ρ          | 0.03   |
| Dividend yield                            | q          | 0      |
| Variance volatility                       | σ          | 0.25   |
| Mean reversal rate                        | κ          | 2      |
| Mean level of variance                    | θ          | 0.04   |
| Price of strike                           | E          | 100    |
| Rate of jump arrival                      | λ          | 0.2    |
| Time to expiry                            | T          | 0.5    |
| Jump size log–variance                    | σ          | 0.4    |
| Jump size log–mean                        | γ          | -0.5   |

Figure 2. Results based on GRBF–FDI in Problem 4.2. Top left: Numerical solution. Top right: List plot of the numerical solution indicating the non-uniform nodes’ distribution. Bottom left: Contour plot of the solution. Bottom right: The sparsity pattern of the system of ODEs’ coefficient matrix.

Numerical comparisons for this case are brought forward in Table 2, once again indicating the fast convergence behavior of the new GRBF–FDI scheme for solving the Bates model. The numerical results have also been illustrated in Figure 2.
| Scheme    | $m$ | $n$ | $N$   | $k$ | $Re(\lambda_{\text{max}})$ | $\epsilon$ | Time (s) |
|-----------|-----|-----|-------|-----|-----------------------------|------------|---------|
| SFD–EM    | 10  | 10  | 100   | 250 | -104.46                     | 4.321 × 10^{-1} | 0.10    |
|           | 15  | 15  | 225   | 500 | -276.31                     | 8.815 × 10^{-2} | 0.23    |
|           | 25  | 20  | 500   | 1000| -884.68                     | 8.121 × 10^{-2} | 0.74    |
|           | 30  | 30  | 900   | 2000| -1361.01                    | 1.498 × 10^{-2} | 1.78    |
|           | 45  | 30  | 1350  | 2000| -3209.72                    | 4.817 × 10^{-3} | 3.77    |
|           | 60  | 30  | 1800  | 5000| -1361.01                    | 1.834 × 10^{-3} | 7.50    |
|           | 80  | 30  | 2400  | 5000| -10982.70                   | 1.434 × 10^{-3} | 13.92   |
| GRBF–FDI  | 10  | 10  | 100   | NR  | -158.68                     | 5.584 × 10^{-2} | 0.16    |
|           | 15  | 15  | 225   | NR  | -435.54                     | 2.018 × 10^{-2} | 0.30    |
|           | 25  | 20  | 500   | NR  | -1381.41                    | 6.922 × 10^{-3} | 0.89    |
|           | 30  | 30  | 900   | NR  | -2054.73                    | 5.998 × 10^{-3} | 1.77    |
|           | 45  | 30  | 1350  | NR  | -4874.35                    | 2.042 × 10^{-3} | 4.26    |
|           | 60  | 30  | 1800  | NR  | -8904.53                    | 1.055 × 10^{-3} | 8.26    |
|           | 80  | 30  | 2400  | NR  | -16165.60                   | 1.434 × 10^{-3} | 13.92   |

Table 4. Numerical reports of put option pricing in Problem 4.3.

| $m$ | $n$ | $\epsilon$ at $s = 90$ | $\epsilon$ at $s = 100$ | $\epsilon$ at $s = 110$ | $\epsilon$ |
|-----|-----|------------------------|--------------------------|--------------------------|------------|
| 17  | 9   | 1.081 × 10^9           | 1.577 × 10^9             | 1.968 × 10^{-1}         | 1.512 × 10^{-1} |
| 33  | 17  | 2.808 × 10^{-2}        | 5.205 × 10^{-1}         | 1.389 × 10^{-1}         | 4.948 × 10^{-2} |
| 65  | 33  | 4.783 × 10^{-3}        | 1.253 × 10^{-1}         | 2.845 × 10^{-2}         | 1.166 × 10^{-2} |
| 129 | 65  | 7.383 × 10^{-3}        | 3.098 × 10^{-2}         | 5.255 × 10^{-3}         | 2.834 × 10^{-3} |
| 257 | 129 | 1.700 × 10^{-5}        | 7.781 × 10^{-3}         | 3.455 × 10^{-3}         | 8.313 × 10^{-4} |

Table 5. Numerical results of the AMG–STS method for Problem 4.3.

**Problem 4.3.** [36] For this experiment, we apply the settings of Table 3 for a put option. The reference values at $s = \{90, 100, 110\}$ and $v = 0.04$ are $11.302917$, $6.589881$, and $4.191455$, respectively.

For this scenario, we have gathered up all the results in Tables 4–5. Figure 3 also illustrates the positivity and smoothness of the numerical solution at the presence of jumps in the model.

Numerical experiment have been provided to verify the discussions given in Sections 1–3 and to show the convergence behavior. In this case the system matrix is a bit dense due to imposing the non–local integral term.

5. **Conclusions.** The importance of pricing under jump–diffusion models has led to the development of several different strategies, most frequently of numerical nature. On top of that, with the advancement of computer hardware and software, the problem of solving PIDE–based models by numerical methods has gained an additional importance.

In this paper and in order to attain high convergence speeds, a competitive GRBF–FD scheme was proposed. Applying the MOL technique, the PIDE was semi–discretized with the GRBF–FD method by taking into consideration a non–uniform grid of nodes with a concentration on the areas of discontinuity and degeneracy.
To tackle the semi-discretized set of ODEs, it is too expensive to compute $e^{\tilde{Y}}$ first and then to post-multiply it by the payoff if $\tilde{Y}$ is large. To remedy this, a Krylov subspace method was applied.

Numerical reports and comparison for our numerical implementation to solve the Bates model in finance confirmed the efficacy and competitiveness of the new presented approach. It preserved the positivity and stability of the computational solution(s) by reducing the computational burdensome.

**Acknowledgment.** We thank a referee for providing helpful comments on an earlier draft of this manuscript.

**Appendix.** Here a Mathematica code is provided as one of the aims of this paper for interested readers when applying the new scheme GRBF–FDI in Problem 4.3. Different parts of the codes have also been described shortly to indicate what that part is doing. One can easily use this code by varying $m$ and $n$, or also modify it for pricing other similar options.

```mathematica
ClearAll["Global`*"];
t1 = AbsoluteTime[];
```
m = 30; n = 30; size = m*n;
Print["The size of the system of ODEs is = ", size];

(***********Problem parameters***********)
TT = 0.5; e = 100.; r = 0.03; q = 0; theta = 0.04; kappa = 2.0;
sigma = 0.25; lambda = 0.2; rho = -0.5; sigma1 = 0.4; mio = -0.5;
zeta = Exp[mio + 1/2 sigma1^2] - 1; omega = r - q - lambda*zeta;

(****Constructing the non-uniform grid****)
xsmin = 0; xsmax = 3 e; ysmin = 0; ysmax = 1.0;
d1 = e/15; d2 = ysmax/150; r1 = 0.25;
sleft = Max[0.5, Exp[-r1*TT]*e]; sright = e;
ksiMin = ArcSinh[(xsmin - sleft)/d1];
kint = (sright - sleft)/d1;
ksiMax = ksiMin + ArcSinh[(xsmax - sright)/d1];
ksi = Range[ksiMin, ksiMax, (ksiMax - ksiMin)/(m - 1)];
fun[ks_] := Which[ksiMin <= ks < 0, sleft + d1*Sinh[ks],
                  0 <= ks <= kint, sleft + d1*ks, kint < ks <= ksiMax, sright + d1*Sinh[ks - kint]];
xn = Threshold@Table[fun[ksi[[i]]], {i, 1, m}];

ny = Table[(j - 1)*del2, {j, 1, n}];
origrid = Flatten[Outer[List, nx, ny], 1];

(***************Filling some matrices***************)
Idx = SparseArray[{{i_, i_} -> 1.}, {m, m}, 0];
xx = SparseArray@DiagonalMatrix@nx;
Idy = SparseArray[{{i_, i_} -> 1.}, {n, n}, 0];
yy = SparseArray@DiagonalMatrix@ny;
X = KroneckerProduct[xx, Idy];
Y = KroneckerProduct[Idx, yy];
Id = KroneckerProduct[Idx, Idy];

(**Differentiation matrix for the first spatial variable**) 
hh = Differences[nx];
ww = Table[hh[[i + 1]]/hh[[i]], {i, 1, Length[hh] - 1}];
c = 3 Max[hh];
A1d1 = Join[Table[{-3 c^2 + hh[[i - 1]]^2 (-5 + 2 hh[[i - 1]])/(3 c^2 hh[[i - 1]] (1 + hh[[i - 1]])),
                  {i, 2, m - 1}],
             {-(1/hh[[Length[hh]]]) + hh[[Length[hh]]]/c^2}];
A1d2 = Join[{-1/hh[[1]]} + hh[[1]]/c^2],
           Table[{-2 hh[[i - 1]] (-1 + hh[[i - 1]])/(3 c^2)}]
\[-(1 + \text{ww}[[i - 1]])/(\text{hh}[[i - 1]] \text{ww}[[i - 1]])\], \{i, 2, m - 1\}, \\
\{1/\text{hh}[[\text{Length[hh]}]]]\};
A1d3 = \text{Join}[[i/\text{hh}[[1]]]]], \\
\text{Table}[\left(3/\text{ww}[[i - 1]] + \left((\text{hh}[[i - 1]])^2 \left(-2 + 5 \text{ww}[[i - 1]]\right)\right)/c^2\right)/
\left(3 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right), \{i, 2, m - 1\}];
A2d1 = \text{Join}[
\text{Table}[
\left(2 \left((3/\text{ww}[[i - 1]] + \left((\text{hh}[[i - 1]])^2 \left(-2 + 5 \text{ww}[[i - 1]]\right)\right)/c^2\right)/
\left(3 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right)\right), \{i, 2, m - 1\}], \\
\{1/\text{hh}[[\text{Length[hh]}]]\}];
A2d2 = \text{Join}[[(-4/c^2), \\
\text{Table}[
\left(2 \left((3/\text{ww}[[i - 1]] + \left((\text{hh}[[i - 1]])^2 \left(-2 + 5 \text{ww}[[i - 1]]\right)\right)/c^2\right)/
\left(3 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right)\right), \{i, 2, m - 1\}], \\
\{2/c^2\}];
A2d3 = \text{Join}[[2/c^2], \\
\text{Table}[
\left(6 \text{c}^2 + 2 \text{hh}[[i - 1]]^2 \left(2 + \text{ww}[[i - 1]]\right) \left(-4 + 5 \text{ww}[[i - 1]]\right)\right)/
\left(3 \text{hh}[[i - 1]]^2 \text{ww}[[i - 1]] (1 + \text{ww}[[i - 1]])\right), \{i, 2, m - 1\}];
dudx = \text{SparseArray}[[\text{Band}[[1, 1]]] \mapsto A1d2, \\
\text{Band}[[2, 1]] \mapsto A1d1, \text{Band}[[1, 2]] \mapsto A2d3, \{m, m\}];
d2udx2 = \text{SparseArray}[[\text{Band}[[1, 1]]] \mapsto A2d2, \\
\text{Band}[[2, 1]] \mapsto A2d1, \text{Band}[[1, 2]] \mapsto A2d3, \{m, m\}];

(*Differentiation matrix for the second spatial variable*)
\text{hh} = \text{Differences[ny]};
\text{ww} = \text{Table}[\text{hh}[[i + 1]]/\text{hh}[[i]], \{i, 1, \text{Length[hh]} - 1\}];
c = 6*\text{Max[hh]};
A1d1 = \text{Join}[
\text{Table}[
\left(-3 \text{cw}^2 + \text{hh}[[i - 1]]^2 \left(-5 + 2 \text{ww}[[i - 1]]\right)\right)/
\left(3 \text{cw}^2 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right), \{i, 2, n - 1\}], \\
\{-1/\text{hh}[[\text{Length[hh]}]]\}, \text{hh}[[\text{Length[hh]}]]/c^2\};
A1d2 = \text{Join}[[(-1/\text{hh}[[1]])) + \text{hh}[[1]]/c^2], \\
\text{Table}[
\left(2 \text{hh}[[i - 1]] \left(-1 + \text{ww}[[i - 1]]\right)/(3 \text{cw}^2)\right) + \left(-1 + \text{ww}[[i - 1]]\right)/
\left(\text{hh}[[i - 1]] \text{ww}[[i - 1]]\right), \{i, 2, n - 1\}, \text{hh}[[\text{Length[hh]}]]\};
A1d3 = \text{Join}[[i/\text{hh}[[1]]]], \\
\text{Table}[\left(3/\text{ww}[[i - 1]] + \left((\text{hh}[[i - 1]])^2 \left(-2 + 5 \text{ww}[[i - 1]]\right)\right)/c^2\right)/
\left(3 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right), \{i, 2, n - 1\}];
A2d1 = \text{Join}[
\text{Table}[
\left(2 \left((3/\text{ww}[[i - 1]] + \left((\text{hh}[[i - 1]])^2 \left(-2 + 5 \text{ww}[[i - 1]]\right)\right)/c^2\right)/
\left(3 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right)\right), \{i, 2, n - 1\}], \\
\{-4/c^2\}];
A2d2 = \text{Join}[[(-4/c^2), \\
\text{Table}[
\left(2 \left((3/\text{ww}[[i - 1]] + \left((\text{hh}[[i - 1]])^2 \left(-2 + 5 \text{ww}[[i - 1]]\right)\right)/c^2\right)/
\left(3 \text{hh}[[i - 1]] (1 + \text{ww}[[i - 1]])\right)\right), \{i, 2, n - 1\}], \\
\{2/c^2\}];
A2d3 = \text{Join}[[2/c^2], 

\[
\begin{align*}
6 \ c^2 + 2 \ hh[[i - 1]]^2 \ (2 + \ ww[[i - 1]] \ (-4 + 5 \ ww[[i - 1]]))/(
3 \ c^2 \ hh[[i - 1]]^2 \ hh[[i - 1]] \ (1 + \ ww[[i - 1]])), \ {i, 2, n - 1});
\end{align*}
\]
\]
dudy = SparseArray[{{Band[{1, 1}] -> A1d2, Band[{2, 1}] -> A1d1, Band[{1, 2}] -> A1d3}, {n, n}}];
d2udy2 = SparseArray[{{Band[{1, 1}] -> A2d2, Band[{2, 1}] -> A2d1, Band[{1, 2}] -> A2d3}, {n, n}}];

(********Deriving the system matrix********)
mat = SparseArray[1/2 Y.(X^2)).KroneckerProduct[d2udx2, Idy] + (rho*sigma)*(X.Y)).(KroneckerProduct[dudx, dudy]) + (1/2 sigma^2 Y).KroneckerProduct[Idx, d2udy2] + (omega*X).(KroneckerProduct[dudx, Idy]) + (kappa*(theta*Id - Y)).(KroneckerProduct[Idx, dudy]) - (r + lambda) Id];

(********Initial condition********)
U[t_] = Flatten@Table[Subscript[u, i, j][t], {i, m}, {j, n}];
payoff = Flatten@Table[Max[e - nx[[i]], 0], {i, m}, {j, n}];

(********Imposing the integral part********)
deltavar = Differences[nx];
A[i_, j_] := N[1/(2 deltavar[[j]]) (E^(mio + sigma1^2/2) (-Erf[(mio + sigma1^2 - N@Log[Rationalize@s[[j]]/s[[i]]])/\(Sqrt[2]\)
 sigma1] + Erf[(mio + sigma1^2 - Log[s[[j + 1]]/s[[i]]])/\(Sqrt[2]\)
 sigma1]) s[[i]] (U[s[[j]], v, t] - U[s[[j + 1]], v, t] + Erf[(mio - \(\text{Log}[\text{Rationalize}[s[[j]]/s[[i]]])]/\(\text{Sqrt}[2]\) sigma1]) + Erf[(mio - \(\text{Log}[s[[j + 1]]/s[[i]]])]/\(\text{Sqrt}[2]\) sigma1]) (s[[j + 1]] U[s[[j]], v, t] - s[[j]] U[s[[j + 1]], v, t]))/]
s = nx;
int1 = SparseArray@Chop@Simplify@Table[
 Sum[A[i, j], {j, 2, m - 1}], {i, 2, m - 1}];
coef1 = SparseArray@Table[Coefficient[int1[[l]], Join[{U[0, v, t]},
 Table[U[nx[[k]], v, t], {k, 2, m}]]], {l, Length@int1}];
int2 = SparseArray@Join[{ConstantArray[0, m]},
 coef1, {ConstantArray[0, m]}];
int5 = SparseArray[Flatten[int2 Partition[U[t], n]]];
right1 = mat.U[t] + lambda int5;
initc = Thread[U[0] == payoff];
eqns = Thread[D[U[t], t] == right1];

(*********Imposing the boundaries*********)
vec0 = SparseArray[{i_} -> 0, size];
mat01 = Table[vec0, {i, n}];
mat02 = SparseArray@Table[-Last@CoefficientArrays[eqns[[i]], U[t]],
  {i, n + 1, (m - 1) n}];
mat03 = SparseArray@ArrayFlatten[{{mat01}, {mat02}, {mat01}}];
mat = mat03;
mat = SparseArray@Chop[TT*mat, 10^-8];

(*********Finding the solution of the set of ODEs*********)
sol2 = MatrixExp[mat, payoff, Method -> "Krylov"];
list1 = Table[Flatten@{origrid[[i]], sol2[[i]]}, {i, 1, size}];
T12 = Map[Last, list1];
set1 = Table[Flatten@{origrid[[i]], T12[[i]]}, {i, 1, size}];

(*********Interpolating the numerical solution*********)
v0 = 0.04;
f = Interpolation@list1;
t2 = AbsoluteTime[] - t1;
Print["The whole computational time is = ", t2];

(*********Reporting the results*********)
{f[90, v0], f[100, v0], f[110, v0]}
Print["The RMSRE is = ", ScientificForm@Sqrt[
  (1/3) (Abs[(11.302917 - f[90, v0])/11.302917])^2 
  + (1/3) (Abs[(6.589881 - f[100, v0])/6.589881])^2 
  + (1/3) (Abs[(4.191455 - f[110, v0])/4.191455])^2
];

(****Reporting the largest and smallest eigenvalues*****)
{Chop@Eigenvalues[mat, 1], Chop@Eigenvalues[mat, -1]}

REFERENCES
[1] M. L. Abell and J. P. Braselton, Differential Equations with Mathematica, Fourth Edition, Academic Press, USA, 2016.
[2] W. Allegretto, Y. Lin and N. Yan, A posteriori error analysis for FEM of American options, Discrete Contin. Dyn. Syst., Ser. B, 6 (2006), 957–978.
[3] M. Balajewicz and J. Toivanen, Reduced order models for pricing European and American options under stochastic volatility and jump–diffusion models, J. Comput. Sci., 20 (2017), 198–204.
[4] L. V. Ballestra and C. Sgarra, The evaluation of American options in a stochastic volatility model with jumps: an efficient finite element approach, Comput. Math. Appl., 60 (2010), 1571–1590.
[5] L. V. Ballestra and L. Cecere, A fast numerical method to price American options under the Bates model, Comput. Math. Appl., 72 (2016), 1305–1319.
[6] D. Bates, Jumps and stochastic volatility: The exchange rate processes implicit in Deutsche mark options, *Rev. Fin. Stud.*, 9 (1996), 69–107.

[7] V. Bayona, M. Moscoso and M. Kindelan, Gaussian RBF–FD weights and its corresponding local truncation errors, *Eng. Anal. Bound. Elem.*, 36 (2012), 1361–1369.

[8] F. Black and M. Scholes, The pricing of options and corporate liabilities, *J. Polit. Econ.*, 81 (1973), 637–654.

[9] M. Briani, C. La Chioma and R. Natalini, Convergence of numerical schemes for viscosity solutions to integro-differential degenerate parabolic problems arising in financial theory, *Numer. Math.*, 98 (2004), 607–646.

[10] M. Caliari, P. Kandolf, A. Ostermann and S. Rainer, Comparison of software for computing the action of the matrix exponential, *BIT*, 54 (2014), 113–128.

[11] S. S. Clift and P. A. Forsyth, Numerical solution of two asset jump diffusion models for option valuation, *Appl. Numer. Math.*, 58 (2008), 743–782.

[12] 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), 87–112.

[13] E. Ekström and J. Tysk, The Black-Scholes equation in stochastic volatility models, *J. Math. Anal. Appl.*, 368 (2010), 498–507.

[14] M. Fakharany, V. N. Egorova and R. Company, Numerical valuation of two-asset options under jump diffusion models using Gauss-Hermite quadrature, *J. Comput. Appl. Math.*, 330 (2018), 822–834.

[15] G. E. Fasshauer, *Meshfree Approximation Methods with MATLAB*, World Scientific Publishing Co., Singapore, 2007.

[16] R. Feng and J. Duan, High accurate finite differences based on RBF interpolation and its application in solving differential equations, *J. Sci. Comput.*, 76 (2018), 1785–1812.

[17] P. G. Giribone and S. Ligato, Option pricing via radial basis functions: Performance comparison with traditional numerical integration scheme and parameters choice for a reliable pricing, *Int. J. Financ. Eng.*, 2 (2015), 1550018, 30 pp.

[18] W. Hackbusch, *Tensor Spaces and Numerical Tensor Calculus*, Springer, Berlin, 2012.

[19] R. L. Hardy, Theory and applications of the multiquadric-biharmonic method: 20 years of discovery, *Comput. Math. Appl.*, 19 (1990), 163–208.

[20] S. L. Heston, A closed-form solution for options with stochastic volatility with applications to bond and currency options, *Rev. Financ. Stud.*, 6 (1993), 327–343.

[21] N. J. Higham, *Functions of Matrices: Theory and Computation*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2008.

[22] K. J. in ’t Hout and S. Foulon, ADI finite difference schemes for option pricing in the Heston model with correlation, *Int. J. Numer. Anal. Modeling*, 7 (2010), 303–320.

[23] T. Kluge, *Pricing Derivatives in Stochastic Volatility Models Using the Finite Difference Method*, Dipl. thesis, TU Chemnitz, 2002.

[24] T. Kolda and B. W. Bader, Tensor decompositions and applications, *SIAM Rev.*, 51 (2009), 455–500.

[25] S. Kou, A jump diffusion model for option pricing, *Management Sci.*, 48 (2002), 1086–1101.

[26] D. C. Lesmana and S. Wang, A numerical scheme for pricing American options with transaction costs under a jump diffusion process, *J. Ind. Manag. Optim.*, 13 (2017), 1793–1813.

[27] J. Loffeld and M. Tokman, Comparative performance of exponential, implicit, and explicit integrators for stiff systems of ODEs, *J. Comput. Appl. Math.*, 241 (2013), 45–67.

[28] M. D. Marcozzi, S. Choi and C. S. Chen, On the use of boundary conditions for variational formulations arising in financial mathematics, *Appl. Math. Comput.*, 124 (2001), 197–214.

[29] A. Mayo, Methods for the rapid solution of the pricing PIDEs in exponential and Merton models, *J. Comput. Appl. Math.*, 222 (2008), 128–143.

[30] R. Merton, Option pricing when underlying stock returns are discontinuous, *J. Financ. Econom.*, 3 (1976), 125–144.

[31] G. H. Meyer, *The Time-Discrete Method of Lines for Options and Bonds, A PDE Approach*, World Scientific Publishing, USA, 2015.

[32] C. B. Moler and C. F. Van Loan, Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later, *SIAM Rev.*, 45 (2003), 3–49.

[33] M. Mureşan, *Introduction to Mathematica with Applications*, Springer, Switzerland, 2017.

[34] H.-K. Pang and H.-W. Sun, Fast exponential time integration for pricing options in stochastic volatility jump diffusion models, *East Asian J. Appl. Math.*, 4 (2014), 52–68.
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 (2009), 668–678.

S. Salmi, J. Toivanen and L. von Sydow, An IMEX–scheme for pricing options under stochastic volatility models with jumps, *SIAM J. Sci. Comput.*, 36 (2014), 817–834.

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

M. Sofroniou and R. Knapp, *Advanced Numerical Differential Equation Solving in Mathematica*, Wolfram Mathematica, Tutorial Collection, USA, 2008.

V. Stolbunov and P. B. Nair, Sparse radial basis function approximation with spatially variable shape parameters, *Appl. Math. Comput.*, 330 (2018), 170–184.

D. Tangman, A. Gopaul and M. Bhuruth, Exponential time integration and Chebychev discretisation schemes for fast pricing of options, *Appl. Numer. Math.*, 58 (2008), 1309–1319.

I. Tolstykh, On using RBF–based differencing formulas for unstructured and mixed structured–unstructured grid calculations, *Proc. 16th IMACS World Congress*, 228 (2000), 4606–4624.

M. Trott, *The Mathematica Guidebook for Numerics*, Springer, New York, NY, USA, 2006.

C. Van Loan, Computing integrals involving the matrix exponential, *IEEE Trans. Autom. Control*, 23 (1978), 395–404.

L. von Sydow, J. Toivanen and C. Zhang, Adaptive finite differences and IMEX time–stepping to price options under Bates model, *Int. J. Comput. Math.*, 92 (2015), 2515–2529.

Received May 2018; revised July 2018.

E-mail address: fazlollah.soleymani@gmail.com
E-mail address: aliakgul100727@gmail.com, aliakgul@siirt.edu.tr