Adaptive finite differences to price European options under the Bates model

Alexander Sjöberg
Abstract

Adaptive finite differences to price European options under the Bates model

Alexander Sjöberg

This thesis presents the pricing of European options under the Bates model, using adaptivity in order to efficiently distribute the grid points in space. For a fixed number of grid points the size of the absolute error, when using the adaptive approach, is reduced compared to the corresponding equidistant grid. Since the adaptive method needs less grid points for a certain error, the linear system of equations that needs to be solved becomes smaller and the memory costs are reduced. The implementation does not rest upon heavy optimization or parallelization theory, but nevertheless it solves the problem flawlessly and the adaptive method outperforms the equidistant method regarding computational time when keeping the error at a predefined level.
1 Introduction

An option is a precise contract between two parties where the holder possesses the right, but not the obligation, to buy (call option) or sell (put option) an underlying instrument or asset at a given strike price $K$ on or before a specified expiry date $T$, also referred to as time of maturity. If the owner decides to exercise the option within the expiry date, the seller, commonly called the writer, has the obligation to sell or buy the underlying asset to the price $K$. The two most common types of options are American options and European options. For an American option the contract can be exercised at any time within the time of maturity as opposed to the European option which can only be exercised on the expiry date.

When considering a European call option, the holder has the alternative to either exercise the option or to let it expire. Which decision the holder makes depends on the price of the underlying asset $X$, called the spot price. If the spot price is bigger than the strike price at time $T$, i.e. $X > K$ then the option is exercised, otherwise not. The value of the option $V$ can be expressed as

$$
\Phi(X) = V(X_T, T) = \max(X_T - K, 0), \tag{1}
$$

where $\Phi(X)$ is the payoff function. When $X > K$ the profit per share for the holder is $X - K$. In analogy, the European put option is only exercised by the holder when $X < K$,

$$
\Phi(X) = V(X_T, T) = \max(K - X_T, 0). \tag{2}
$$

The option trading has grown tremendously during the last couple of decades and due to the prevailing situation, the demand of models that price options is high. There are many different models for pricing options of varying complexity. The more complicated models reflect the paths for the value more realistically compared to a simple model and the match between the market price and the modeled price of the option is more accurate. One of the most commonly used models is the Black-Scholes model. This model is described by the stochastic differential equation

$$
dX_t = \mu X_t dt + \sigma X_t dW_t, \tag{3}
$$

where $X_t$ is the price of the underlying asset, $\mu$ is the drift rate of $X$ and $\sigma$ is the volatility of $X$. This model was introduced in 1973 by Black and Scholes in [2] and Merton in [4].

A few years later Merton presented another model [5] that adds log-normally distributed jumps to the Black-Scholes model. These jumps imply that sudden changes in the underlying asset are taken into account.

The Heston model, see [3], is an option pricing model that allows stochastic variations in the volatility in the Black-Scholes model.
This paper considers the pricing of European options using the Bates model [1], which combines Heston’s stochastic volatility model with the Merton jump model. The model describes the behavior of the asset value $X$ and its variance $Y$ by the coupled stochastic differential equations

$$
\begin{align*}
  dX_t &= (r - q - \lambda \xi)X_t \, dt + \sqrt{Y_t} \, X_t \, dW^1_t + (J - 1) \, X_t \, dn,
  \\
  dY_t &= \kappa(\theta - Y_t) \, dt + \sigma \sqrt{Y_t} \, dW^2_t,
\end{align*}
$$

where $r$ and $q$ are the risk free interest rate and the dividend yield respectively, $\lambda$ is the intensity for the Poisson process $N$, $\xi$ is the mean jump, $W^1$ and $W^2$ are Wiener processes with correlation $\rho$, $\kappa$ is the rate of reversion to the mean level of the variance $Y$, $\theta$ is the mean level of $Y$ and $\sigma$ is the volatility of $Y$. The jump size of the Poisson process is denoted by $J$ and has the distribution

$$
f(J) = \frac{1}{\sqrt{2\pi \delta J}} \exp\left(-\frac{[\ln J - (\gamma - \delta^2/2)]^2}{2\delta^2}\right).
$$

Here $\gamma$ and $\delta$ describe the mean of the jump and the variance of the jump respectively. The relationship between $\gamma$ and $\xi$ is $\gamma = e^{\xi} - 1$. Let $\tau = T - t$ denote the time to maturity and consider the partial integro-differential equation

$$
\frac{\partial u(x, y, \tau)}{\partial \tau} = \frac{1}{2} y x^2 \frac{\partial^2 u(x, y, \tau)}{\partial x^2} + \rho \sigma y x \frac{\partial^2 u(x, y, \tau)}{\partial x \partial y} + \frac{1}{2} \sigma^2 y \frac{\partial^2 u(x, y, \tau)}{\partial y^2} + (r - q - \lambda \xi) x \frac{\partial u(x, y, \tau)}{\partial x} + \kappa(\theta - y) \frac{\partial u(x, y, \tau)}{\partial y} - (r + \lambda) u(x, y, \tau)
\quad + \lambda \int_0^\infty u(Jx, y, \tau) f(J) \, dJ,
$$

where $x$ is the value of the asset and $y$ is the volatility. By introducing the operator

$$
\mathcal{L}_P\mathcal{D}_E = \frac{1}{2} y x^2 \frac{\partial^2}{\partial x^2} + \rho \sigma y x \frac{\partial^2}{\partial x \partial y} + \frac{1}{2} \sigma^2 y \frac{\partial^2}{\partial y^2} + (r - q - \lambda \xi) x \frac{\partial}{\partial x} + \kappa(\theta - y) \frac{\partial}{\partial y} - (r + \lambda),
$$

the partial integro-differential equation in (6) can be written as

$$
\frac{\partial u(x, y, \tau)}{\partial \tau} = \mathcal{L}_P\mathcal{D}_E u(x, y, \tau) + \lambda \int_0^\infty u(Jx, y, \tau) f(J) \, dJ.
$$

For computational reasons, the unbounded domain is truncated,

$$
x \in (0, X), \quad y \in [0, Y], \quad \tau \in (0, T).
$$
The payoff function $\Phi(x, y)$ defines the initial values for $u$ and gives the value of the option at the expiry date. The boundary conditions that are used for solving (6) are

\begin{align*}
  u(0, y, \tau) &= \Phi(0, y) \\
  u(X, y, \tau) &= \Phi(X, y) \\
  \frac{\partial u(x, y, \tau)}{\partial y} &= 0.
\end{align*}

For the case where $y = 0$ no boundary condition is needed. One-sided differences are used for discretization in this area.

Figure 1: Illustration of the two-dimensional grid. The circles that are not filled denote excluded points and filled circles are included points.

\section{Space discretization}

When using adaptivity, the objective is to increase the number of grid points where needed and have a more sparse grid in areas where not much happens in the solution. By keeping track of the truncation error one can vary the step lengths and thereby generate an adaptive grid. The approach used in this article is

1. Generate a fixed, coarse grid and solve the problem.

2. Create a new grid with variable step length for higher accuracy. These variable step lengths are based on the estimated truncation error achieved from the first solution.

The problem needs to be solved two times, but solving it with the coarse grid is supposed to take little time with low accuracy, but will nevertheless tell how to place the grid points for the second solution. Figure 1 illustrates the grid, where included points are denoted by • and excluded points are shown as o.

2.1 Space discretization with fixed step lengths

Let $\alpha_1, \alpha_2, \ldots, \alpha_6$ denote the coefficients in (6),

\[
\begin{align*}
\alpha_1 &= \frac{1}{2} y x^2, & \alpha_2 &= \frac{1}{2} \sigma^2 y, & \alpha_3 &= \rho \sigma x y, \\
\alpha_4 &= (r - q - \lambda \xi) x, & \alpha_5 &= \kappa (\theta - y), & \alpha_6 &= -(r + \lambda).
\end{align*}
\]

The PDE part becomes

\[L_{\text{PDE}} u = \alpha_1 u_{xx} + \alpha_2 u_{yy} + \alpha_3 u_{xy} + \alpha_4 u_x + \alpha_5 u_y + \alpha_6 u \quad (9)\]

and these derivatives of $u$ need to be discretized, using the standard second order central difference approximations. Introduce the semi-discrete vector function $i,j(\tau) = u(x_i, y_j, \tau)$, for $1 \leq i \leq n, 1 \leq j \leq m$ and step lengths $h_x = \frac{X}{n}, h_y = \frac{Y}{m}$

\[
\begin{align*}
    u_{xx} &\approx \frac{\psi_{i+1,j}(\tau) - 2\psi_{i,j}(\tau) + \psi_{i-1,j}(\tau)}{h_x^2}, \\
    u_{yy} &\approx \frac{\psi_{i,j+1}(\tau) - 2\psi_{i,j}(\tau) + \psi_{i,j-1}(\tau)}{h_y^2}, \\
    u_{xy} &\approx \frac{\psi_{i+1,j+1}(\tau) - \psi_{i+1,j-1}(\tau) - \psi_{i-1,j+1}(\tau) + \psi_{i-1,j-1}(\tau)}{4h_x h_y}, \\
    u_x &\approx \frac{\psi_{i+1,j}(\tau) - \psi_{i-1,j}(\tau)}{2h_x}, \\
    u_y &\approx \frac{\psi_{i,j+1}(\tau) - \psi_{i,j+1}(\tau)}{2h_y}.
\end{align*}
\]

By applying the approximations in (10) on the PDE, a 9-point discretization stencil is obtained for each $\psi_{i,j}(\tau)$, see Figure 2.

Introduce the coefficient matrix $A_h$ which is the second order discretisation of $L_{\text{PDE}}$ and is used in the linear equation system that is later to be solved. From the boundary conditions in the $x$-direction, known elements are obtained which result in a column vector $a$. The vector $\psi$ is the lexicographic ordered column vector

\[\psi = (\psi_{1,1}, \psi_{1,2}, \ldots, \psi_{j,i}, \ldots, \psi_{m,n})^T.\]

The marked grid point $(i,j)$ in Figure 1 corresponds to the element $l = (j-1)n + i$ in $\psi$. The coefficient matrix $A_h$ is a $nm \times nm$ matrix with the
Figure 2: 9-point discretization stencil.

following elements for the interior points $2 \leq i \leq n - 1$ and $2 \leq j \leq m - 1$:

\[
\begin{align*}
A_h(l, l - 1 + n) &= -\alpha_3 \frac{1}{4h_x h_y}, & A_h(l, l + n) &= \alpha_2 \frac{1}{h_y} + \alpha_5 \frac{1}{2h_y}, \\
A_h(l, l + 1 + n) &= \alpha_3 \frac{1}{4h_x h_y}, & A_h(l, l - 1) &= \alpha_1 \frac{1}{h_x^2} - \alpha_4 \frac{1}{2h_x}, \\
A_h(l, l) &= \alpha_6 - \alpha_1 \frac{2}{h_x^2} - \alpha_2 \frac{2}{h_y^2}, & A_h(l, l + 1) &= \alpha_1 \frac{1}{h_x^2} + \alpha_4 \frac{1}{2h_x}, \\
A_h(l, l - 1 - n) &= \alpha_3 \frac{1}{4h_x h_y}, & A_h(l, l - n) &= \alpha_2 \frac{1}{h_y^2} - \alpha_5 \frac{1}{2h_y}, \\
A_h(l, l + 1 - n) &= -\alpha_3 \frac{1}{4h_x h_y} r
\end{align*}
\]

2.2 Space discretization with variable step lengths

When solving the problem a second time with varying step sizes the derivative approximations of $u$ become quite different. The first derivatives with respect to $x$ and $y$ respectively are approximated with

\[
\begin{align*}
u_x &\approx a_{x,i} \psi_{i+1,j}(\tau) + b_{x,i} \psi_{i,j}(\tau) + c_{x,i} \psi_{i-1,j}(\tau), \\
u_y &\approx a_{y,i} \psi_{i,j+1}(\tau) + b_{y,i} \psi_{i,j}(\tau) + c_{y,i} \psi_{i,j-1}(\tau),
\end{align*}
\]

(11)
When introducing the change of variable $x$

The integral term in (7) needs to be evaluated in each grid point $0 = x_0, x_1, x_2, \ldots, x_i, \ldots, x_n, x_{n+1} = X_{\text{max}}$ and the notation used for this term is

$$I_i = \int_0^\infty u(Jx_i, y, \tau) f(J) \, dJ.$$

(14)

When introducing the change of variable $J = e^s$, (14) becomes

$$I_i = \int_{-\infty}^{\infty} e^s u(e^s x_i, y, \tau) f(e^s) \, ds.$$

(15)

When decomposing the integral in (15), it can be written as

$$I_i = \sum_{k=0}^{n} I_{i,k} + \int_{ln x_{\text{max}}}^{\infty} \Phi(e^s x_i, y)p(s) \, ds,$$

(16)

where

$$a_{x_i} = \frac{h_{x_i}}{h_{x_i}^+(h_{x_i}^{-} + h_{x_i}^+)}$$

$$a_{y_j} = \frac{h_{y_j}^- (h_{y_j}^+ + h_{y_j}^+)}{h_{y_j}^+}$$

$$b_{x_i} = \frac{h_{x_i}^+ - h_{x_i}^-}{h_{x_i}^+ h_{x_i}^-}$$

$$b_{y_j} = \frac{h_{y_j}^+ - h_{y_j}^-}{h_{y_j}^+ h_{y_j}^-}$$

$$c_{x_i} = -\frac{h_{x_i}^+}{h_{x_i}^+ (h_{x_i}^+ + h_{x_i}^-)}$$

$$c_{y_j} = -\frac{h_{y_j}^+}{h_{y_j}^+ (h_{y_j}^+ + h_{y_j}^-)}.$$

Here $h_{x_i}^+ = x_{i+1} - x_i$, $h_{x_i}^- = x_i - x_{i-1}$, $h_{y_j}^+ = y_j + 1 - y_j$ and $h_{y_j}^- = y_j - y_{j-1}$.

Approximations used for the second derivatives are

$$u_{xx} \approx a_{x_i,x_i} \psi_{i+1,j}(\tau) + b_{x_i,x_i} \psi_{i,j}(\tau) + c_{x_i,x_i} \psi_{i-1,j}(\tau),$$

$$u_{yy} \approx a_{y_j,y_j} \psi_{i,j+1}(\tau) + b_{y_j,y_j} \psi_{i,j}(\tau) + c_{y_j,y_j} \psi_{i,j-1}(\tau),$$

(12)

where

$$a_{x_i,x_i} = \frac{2}{h_{x_i}^+(h_{x_i}^- + h_{x_i}^+)}$$

$$a_{y_j,y_j} = \frac{2}{h_{y_j}^+(h_{y_j}^- + h_{y_j}^+)}$$

$$b_{x_i,x_i} = -\frac{2}{h_{x_i}^+ h_{x_i}^-}$$

$$b_{y_j,y_j} = -\frac{2}{h_{y_j}^+ h_{y_j}^-}$$

$$c_{x_i,x_i} = \frac{2}{h_{x_i}^+ (h_{x_i}^- + h_{x_i}^+)}$$

$$c_{y_j,y_j} = \frac{2}{h_{y_j}^+ (h_{y_j}^- + h_{y_j}^+)}$$

and the mixed term becomes

$$u_{xy} \approx a_{y_j,x_i} u_{x_i}(x_i, y_{j+1}, \tau) + b_{y_j,x_i} u_{x_i}(x_i, y_j, \tau) + c_{y_j,x_i} u_{x_i}(x_i, y_{j-1}, \tau)$$

$$\approx a_{y_j,x_i} (a_{x_i,x_i,1} u_{x_i,j+1} + b_{x_i,x_i,j+1} + c_{x_i,x_i,1,j+1})$$

$$+ b_{y_j,x_i} (a_{x_i,x_i,j} + b_{x_i,x_i,j} + c_{x_i,x_i,j})$$

$$+ c_{y_j,x_i} (a_{x_i,x_i,j-1} + b_{x_i,x_i,j-1} + c_{x_i,x_i,j-1}).$$

(13)

The same approximations are used and discussed more exhaustively in [6].

### 2.3 Discretization of the integral term

The integral term in (7) needs to be evaluated in each grid point $0 = x_0, x_1, x_2, \ldots, x_i, \ldots, x_n, x_{n+1} = X_{\text{max}}$ and the notation used for this term is

$$I_i = \int_0^\infty u(Jx_i, y, \tau) f(J) \, dJ.$$

(14)

When introducing the change of variable $J = e^s$, (14) becomes

$$I_i = \int_{-\infty}^{\infty} e^s u(e^s x_i, y, \tau) f(e^s) \, ds.$$

(15)

When decomposing the integral in (15), it can be written as

$$I_i = \sum_{k=0}^{n} I_{i,k} + \int_{ln x_{\text{max}}}^{\infty} \Phi(e^s x_i, y)p(s) \, ds,$$

(16)
where \( p(s) = e^s f(e^s) \) and

\[
I_{i,k} = \int_{\ln \frac{x_k}{x_i}}^{\ln \frac{x_{k+1}}{x_i}} u(e^s x_i) p(s) \, ds.
\]

By using the trapezoidal rule, the integral \( I_{i,k} \) approximates to

\[
I_{i,k} \approx \frac{1}{2} \ln \frac{x_{k+1}}{x_k} \left[ u(e^{\ln \frac{x_k}{x_i}} x_i, y) p(\ln \frac{x_k}{x_i}) + u(e^{\ln \frac{x_{k+1}}{x_i}} x_i, y, \tau) p(\ln \frac{x_{k+1}}{x_i}) \right]
\]

\[
= \frac{1}{2} \ln \frac{x_{k+1}}{x_k} \left[ u(x_k, y, \tau) p(\ln \frac{x_k}{x_i}) + u(x_{k+1}, y, \tau) p(\ln \frac{x_{k+1}}{x_i}) \right]
\]

The integral \( I_{i,k} \) has to be evaluated differently for \( k=0 \) since \( \ln(x_0/x_i) = \ln(0) = -\infty \),

\[
I_{i,0} = \int_{-\infty}^{\ln \frac{x_1}{x_i}} u(e^s x_i, y, \tau) p(s) \, ds \approx u(x_i, y, \tau) \int_{-\infty}^{\ln \frac{x_1}{x_i}} p(s) \, ds
\]

\[
= \frac{1}{\sqrt{2\pi\delta}} u(x_i, y, \tau) \int_{-\infty}^{\ln \frac{x_1}{x_i}} e^{-\frac{(s-(\gamma-\delta^2/2))^2}{2\delta^2}} \, ds
\]

\[
= u(x_i, y, \tau) \frac{1}{\sqrt{\pi}} \int_{-\hat{z}}^{\infty} e^{-z^2} \, dz = \frac{1}{2} u(x_i, y, \tau) \text{erfc}(-\hat{z}),
\]

where \( \text{erfc}(\cdot) \) is the complementary error function and \( \hat{z} = -\frac{\ln \frac{x_1}{x_i}-(\gamma-\delta^2/2)^2}{2\delta^2} \).

The second part in (16) is still to be evaluated. To solve this part one uses the approximations \( \text{erf}(\infty - (\gamma - \delta^2/2)) \approx -1 \) and \( \text{erf}(\gamma + \delta^2/2 - \infty) \approx 1 \). Note that \( e^s x_i > K \) for all \( s \) since the lower boundary for the integral is \( e^{\ln X_{max}/x_i} x_i = X_{max} > K \). Hence,

\[
\int_{\ln \frac{X_{max}}{x_i}}^{\infty} \Phi(e^s x_i, y) p(s) \, ds
\]

\[
= \frac{1}{\sqrt{2\pi\delta}} \int_{\ln \frac{X_{max}}{x_i}}^{\infty} \max(e^s x_i - Ke^{-\tau}, 0) e^{-\frac{(s-(\gamma+\delta^2/2))^2}{2\delta^2}} \, ds
\]

\[
= \frac{1}{2} e^{\gamma} \left[ 1 + \text{erf}(\frac{\gamma + \delta^2/2 - \ln \frac{X_{max}}{x_i}}{\sqrt{2}\delta}) \right]
\]

\[
+ \frac{1}{2} Ke^{-\tau} \left[ 1 + \text{erf}(\frac{\ln \frac{X_{max}}{x_i} - (\gamma - \delta^2/2)}{\sqrt{2}\delta}) \right].
\]

The result of the discretization of the integral term is a \( nm \times nm \) matrix \( I_h \) with full block matrices of size \( n \times n \) on the diagonal and a vector \( \mathbf{b} \) obtained from (17). A similar approach for discretizing the integral is presented in [9].
3 Adaptivity

This thesis only considers adaptivity for the space dimension $x$. The approach used here is the same as the one presented in [6]. The idea with adaptivity is to control the local discretization error $\tau_h$ in order to vary the step sizes in space and create a grid in which the grid points are distributed efficiently. Consider the solution $u(x,y)$. It holds that

$$A_h x y \psi_{h x y} = Au + \tau_{h x y},$$

where $\psi_{h x y}$ is the vector of unknown and $A_{h x y}$ is the discrete approximation of the operator $PDE$ as before. Use the approximation

$$\tau_{h x y} \approx h_x^2 \eta_x(x,y) + h_y^2 \eta_y(x,y) = \tau_{h x} + \tau_{h y},$$

and define

$$\delta_{h x} = Au + \tau_{h x},$$
$$\delta_{2h x} = Au + \tau_{2h x}.$$  

The local truncation error of the second order approximation on the grid with step size $h$ can be approximated with the coarse grid with step size $2h$

$$\tau_{h x} = \frac{1}{2^2 - 1} (\delta_{2h x} - \delta_{h x}).$$

These calculations are done for several points in time and the maximum absolute value of $\tau_{h x}$ is used to calculate the new grid. To approximate $\eta_x(x,y)$ one computes a solution when using step size $\bar{h}_x$ and then create a new matrix $A_{2\bar{h}_x}$ corresponding to the grid with step size $2\bar{h}$. By calculating $\tau_{h x}$ as in (19), $\eta_x(x,y)$ is obtained,

$$\eta_x(x,y) = \frac{\tau_{h x}(x,y)}{h_x^2}.$$  

In order to control the local discretization error, a tolerance $\epsilon$ is introduced, such that $|\tau_{h x}(x,y)| \leq \epsilon_x$ for any $\epsilon_x > 0$. Combining this tolerance with equations (18) and (20) one can write

$$|\tau_{h x}(x)| = |h_x^2(x)\eta_x(x,y)| \approx \left| h_x^2(x) \frac{\tau_{h x}(x,y)}{h_x^2(x)} \right| \leq \epsilon_x.$$

This way one obtain the discrete function $h_x(x,y)$ that gives the varying steps for the adaptive grid. Take the maximal absolute value of $\tau_{h x}(x,y)$ over the $y$-dimension,

$$\hat{\tau}_{h x}(x) = \max_y |\tau_{h x}(x,y)|.$$
Substituting $\tau_{hx}(x, y)$ with $\hat{\tau}_{hx}(x)$ in (21) one obtain

$$h_x(x) = \overline{h}_x^2(x) \left( \frac{e_x}{\overline{\tau}_{hx}(x)} \right)^{\frac{1}{2}},$$

(22)

A weighted mean-value filter is used a number of times to ensure a smooth $\hat{\tau}_{hx}$,

$$\hat{\tau}_{hx}(x_i) = (\hat{\tau}_{hx}(x_{i-1}) + 2\hat{\tau}_{hx}(x_i) + \hat{\tau}_{hx}(x_{i+1}))/4.$$

To prevent the function $h_x(x)$ from taking too large steps when the local truncation error is very small, a parameter $\gamma$ is introduced and the expression in (22) becomes

$$h_x(x) = \overline{h}_x^2(x) \left( \frac{e_x}{e_x \gamma + \overline{\tau}_{hx}(x)} \right)^{\frac{1}{2}}.$$

4 Time discretization

To solve the linear system of equations that is obtained after discretizing (7), the backward differentiation formula BDF-2 has been used for all time steps except the first one, since BDF-2 is an implicit multi-step method. Euler backwards is used for the first time step,

$$\psi^k = \Delta t (A + I) \psi^k + \Delta t (a + b) + \psi^{k-1},$$

where $k = 1$ and $\Delta t$ is the step size in time. For each $k = 2, 3, \ldots, \frac{T}{\Delta t}$ the system of equations obtained from the BDF-2 needs to be solved,

$$\frac{3}{2} \psi^k = \Delta t (A + I) \psi^k + \Delta t (a + b) + 2 \psi^{k-1} - \frac{1}{2} \psi^{k-2}.$$

5 Implementations

The implementation has been made in MATLAB and the matrix $A$ has been generated using MATLAB’s sparse allocation command spalloc. The iterative method Generalized Minimal Residual Method, or GMRES, see [8], has been used for solving the large, linear, sparse system of equations obtained from the implicit methods Euler backward and BDF-2. For efficiency reasons GMRES restarts after six iterations. The iterations are stopped when the relative residual norm is small enough, here chosen to $10^{-8}$. MATLAB’s built-in, sparse, incomplete LU factorization, ilu has been used as a preconditioner. The factors $L$ and $U$ only needs to be calculated once for Euler backwards and once more for BDF-2.

All the experiments presented in Section 6 were run on a computer with the following properties

- CPU: AMD Opteron (Bulldozer) 6274, 2.2 GHz, 16-cores, dual socket,
• Memory: 120 GB,
• Operating system: Scientific Linux 6.3.

Even though the computer has 32 cores the speed-up compared to running the program on a dual core computer is very small. The reason for this is that nothing in the program has been parallelized, since the parallelization overheads in MATLAB’s Parallel Computing Toolbox are too big and no significant speed-up would be achieved. If the code was implemented in Fortran or C++ and parallelized, the performance would improve in comparison to the current implementation.

6 Results

Numerical experiments have been made to compare the method that price European call options under the Bates model using finite differences and adaptivity, with the corresponding equidistant method. Consider the risk free interest rate \( r = 0.03 \), the dividend yield \( q = 0.05 \), the strike price \( K = 100 \), the correlation between the two Wiener processes \( \rho = -0.5 \), the mean level of the variance \( \theta = 0.04 \), the rate of reversion \( \kappa = 2.0 \), the volatility of the variance \( \sigma = 0.25 \), the intensity rate of the jump \( \lambda = 0.2 \), the mean jump \( \gamma = -0.5 \) and the variance of the jump \( \delta = 0.4 \).

The computational domain used for these experiments is

\[
x \in (0, X_{\text{max}}), \quad y \in [0, Y], \quad \tau \in (0, T).
\]

Here \( X_{\text{max}} \) is four times the strike price \( K \), which is a commonly used truncation when modeling option prices. This rule of thumb is also used in [6] and [7]. In reality the price of the underlying asset will, in most cases, be in the region of the strike price. Considering this, the error measurement used for these experiments is the financial error, discussed in [7],

\[
E = \max_{(x,y) \in \Omega_K} |u(x,y,T) - \psi(x,y,T)|
\]  

(23)

where

\[
\Omega_K = \{x|K - \frac{2K}{3} \leq x \leq K + \frac{2K}{3}, y|0.1 \leq y \leq 0.9\}.
\]

A reference solution has been produced by solving the problem with a fine, equidistant grid with 1022 points in the \( x \)-dimension, 512 points in the \( y \)-dimension and 256 points in time. This solution has been considered to be a good estimate for \( u(x,y,T) \) in (23).

The problem has been solved for several different computational grids. The number of grid points in space have varied for both dimensions, while the number of time steps have been kept constant, i.e \( k = 256 \). The ratio
between \( n \) and \( m \) is \( n = 2m - 2 \). The first and the last point in \( x \), i.e \( X_{\text{min}} \) and \( X_{\text{max}} \) respectively, are not included in the grid.

The parameter \( \gamma_x \) that prevents the function \( h_x(x) \) from taking too big steps in the space dimensions when using adaptivity, is set to 0.01. The parameter \( \epsilon_x \) is adjusted in order to get the desired number of grid points in the adaptive space grid. To create the adaptive grid, a coarse grid is first generated and the problem is then solved with the coarse grid, as mentioned in Section 2. Regardless of the number of grid points in the fine, adaptive grid, the same coarse grid have been used, namely \((n, m, k) = (40, 40, 40)\). Figure 3 is an example of the solution when solving the problem using an adaptive grid.

![Adaptive grid using \((n, m, k) = (40, 24, 256)\)](image)

Figure 3: Solution using an adaptive grid with 46 points in the \( x \)-direction and 24 points in the \( y \)-direction.

The pricing of European call options under the Bates model, with the same parameter values as defined in this article, has been done by Jari Toivanen, Stanford University, and the results from his experiments are compared with the ones presented here, see Table 1. As opposed to this article, Toivanen used a component wise splitting method for his computations. The column in Table 1 that is labeled error shows the absolute value of the difference between Toivanen’s calculated prices compared to the prices obtained using finite differences discussed in this report. The volatility for these experiments has been chosen to \( y = 0.04 \) and the value of the option has been calculated for five different values of \( x \), i.e \( x = (80, 90, 100, 110, 120) \). Note
that all elements in $x$ are in a close region of $K$. The corresponding option prices that Jari Toivanen obtained from his experiments are

$$u = (0.32844533, 2.10886115, 6.71067452, 13.74712019, 22.13278777).$$

For most cases, the error in Table 1 decreases when increasing the number of grid points. In Figure 4 and Figure 5, a comparison between the adaptive method and the method using equidistant grid is made. The maximal absolute error, defined in (23), is plotted as a function of total number of grid points in Figure 4 and same is done for the maximal relative error in Figure 5. It is obvious that the discretization error for the adaptive method is smaller compared to the equidistant case, for a given number of points in space. The dashed line in Figures 4, 5 has slope $-1$ which corresponds to the optimal convergence rate, since the $x$-axis in both cases is the total number of grid points, i.e. the number of points in $x \times$ the number of points in $y$.

Furthermore, the CPU time is plotted as a function of the maximal absolute error and the comparison between the adaptive and the equidistant case is once again made and presented in Figure 6. It becomes clear that the adaptive method gives a better accuracy for a certain amount of CPU time, relative the equidistant method. When the error is large though, the equidistant method needs less CPU time than the adaptive one, since the grid points used to generate the solution are very few and the extra time that it takes to solve the problem two times for the adaptive method, discussed in Section 2, has a significant effect on the total execution time.
Figure 4: The size of the greatest absolute error versus number of grid points, comparing the adaptive method and the equidistant method.

Figure 5: The size of the greatest relative error versus number of grid points, comparing the adaptive method and the equidistant method.
Figure 6: Execution time versus the size of the greatest absolute error, comparing the adaptive method and the equidistant method.
7 Conclusion

This thesis describes the pricing of European call options under the Bates model using finite difference discretization and adaptivity in the space dimension $x$, i.e., the dimension corresponding to the price of the underlying asset. The integral that comes with the partial integro-differential equation has been discretized by using the trapezoidal rule.

From the numerical experiments one can conclude that the adaptive method generates a more accurate solution than the equidistant method. The adaptive method needs significantly less grid points for a specific error, compared to the equidistant method. This implies that, for a certain size of the discretization error, a smaller system of equations needs to be solved when considering the adaptive method and therefore the CPU time is reduced. If the implementation was made in a program that is more suited for parallelization than MATLAB, an even greater speedup would have been obtained since there are many parts in the code that can be parallelized.
References


