Option Pricing under a Heston Volatility model using ADI schemes

Jieshun Luo, Qi Wang, Nestor Carbayo

March 12, 2015

1 Introduction

This paper deals with the implementation of an ADI finite difference scheme to solve a two dimensional PDE: the Heston PDE. The two variables in this PDE are the stock price and the (stochastic) volatility.

Under the Heston stochastic volatility model, the stock price dynamics under the risk neutral probability measure $Q$ are given by: [2]

$$dS_t = rS_t dt + \sqrt{V_t}S_t dW^1_Q$$

(1)

Where the volatility, $\sqrt{V_t}$, is itself an stochastic process, rather than a constant as in the original Black-Scholes model, whose dynamics are given by:

$$dV_t = \kappa(\eta - V_t)dt + \sigma \sqrt{V_t}dW^2_Q$$

(2)

Where, $a$, $b$, and $\sigma$ are positive constants and $W^1_Q$ and $W^2_Q$ are two correlated Brownian motions under $Q$ with quadratic co-variation:

$$[W^1_Q, W^2_Q] = \rho t$$

(3)

$\rho \in [-1, 1]$

It can be shown, after introducing the stochastic dynamics of the price of a bond under both the domestic and the foreign currency$^1$ [3], that the risk-neutral price, $u(t, S_t, V_t)$, at time $T-t$ of a $T$-maturity European type option in this stochastic volatility model satisfies the partial differential equation:

$$\frac{\partial u}{\partial t} = \frac{1}{2} \sigma^2 v \frac{\partial^2 u}{\partial s^2} + \rho \sigma v \frac{\partial^2 u}{\partial s \partial v} + \frac{1}{2} \sigma^2 v \frac{\partial^2 u}{\partial v^2} + (r_d - r_f)s \frac{\partial u}{\partial s} + \kappa(\eta - v) \frac{\partial u}{\partial v} - r_d u$$

(4)

with some initial and boundary conditions that depend on the particular option.

In this paper we aim to replicate the splitting schemes of the Alternative Direction Implicit (ADI) type proposed by Hout and Foulon [1] in order to find a numerical solution for the Heston PDE (4). The material presented here relies heavily on the content of that paper.

The schemes proposed by Hout and Foulon in their paper overcomes the difficulty arising from the mixed derivative term, which comes from the correlation between the Brownian Motion driving the asset dynamics and the one driving the stochastic volatility dynamics. Note that if we had $\rho = 0$ in (3) there would be no mixed derivatives term in (4).

---

$^1$The diffusion coefficient used for the bond price dynamics is of the same type as in (1)
2 Analysis of the Heston PDE [1]

2.1 Parameters

The parameters in (4) are: $t$, time to maturity; $v$, variance; $s$, price of the underlying; $\rho \in [-1, 1]$, instantaneous correlation between the Brownian motions driving the underlying price process and the stochastic volatility process; $\sigma > 0$, volatility of the variance; $r_d$, domestic interest rate; $r_f$, foreign interest rate; $\kappa > 0$ the mean reversion rate; $\eta > 0$, with $2\kappa\eta > \sigma^2$, the long term mean.

2.2 Initial and Boundary conditions

We used the boundary conditions proposed in [1]:

Initial condition for a European call option:

$$u(s, v, 0) = \max(0, s - K)$$

Where $K$ is the strike price of the option.

Boundary conditions for European call option, with $(0 \leq t \leq T)$

$$u(0, v, t) = 0$$

$$\lim_{s \to \infty} \frac{\partial u}{\partial s}(s, v, t) = e^{-r_f t}$$

$$\lim_{v \to \infty} u(s, v, t) = se^{-r_f t}$$

We get a further boundary condition at $S$ from the discretization of the PDE described in section 3, $\frac{\partial^2 u}{\partial S^2}(S, v, t)$ is approximated using a central difference scheme, using the virtual point $S + \Delta s_{m_1}$.

Different boundary conditions are proposed in the literature [6]. For example, the boundary conditions proposed in [4] are based on the fact that the option gamma approaches zero for very small or very large asset prices, so $\frac{\partial^2 u}{\partial S^2}(0, v, t) = 0$, $\lim_{s \to \infty} \frac{\partial^2 u}{\partial S^2}(s, v, t) = 0$; and on forcing that (4) is satisfied at $v = 0$, $\frac{\partial u}{\partial t}(s, 0, t) = (r_d - r_f)s \frac{\partial u}{\partial s}(s, 0, t) + \kappa \eta \frac{\partial^2 u}{\partial v^2}(s, 0, t) - r_d u(s, 0, t)$

3 Finite Difference Schemes for the Heston PDE

3.1 Space discretization

3.1.1 Mesh definition in space

To implement a finite difference scheme, we first need to define a grid along the two space variables, $s$ and $v$. We have the choice between uniform and non-uniform grids. Non-uniform grids are more difficult to handle. However, they allow to have a finer partition near the points of more interest. Hence, non-uniform grids are often preferable since they produce more accurate prices with fewer grid points, and consequently, with less computation time.

We apply, as Hout and Foulon [1], a non uniform grid with a higher density of nodes near $s = K$ and $v = 0$. We need a denser grid in this area because the derivative of the initial condition given by (5) has a discontinuity at $K = 0$, which makes the scheme less exact. Furthermore, when $v \approx 0$ the Heston PDE is convection-dominated, which also causes problems. Finally, we would like to have a higher density of points in the neighborhood of $(s, v) = (K, 0)$ since most options are traded with parameter values close to these.

In the $s$-direction we use the following mesh for $0 = s_0 < s_1 < \ldots < s_{m_1} = S$

$$s_i = K + csinh(\xi_i) \quad (0 \leq i \leq m_1)$$

where:
• $m_1 \geq 1$ is an integer defining the number of points (minus one) in the $s$-axis

• $c > 0$ is a constant that controls the fraction of the mesh points $s_i$ that lie around $s = K$. In the implementation we used the same value for $c$ proposed in [1]: $c = \frac{K}{5}$.

• $\xi_0 < \xi_1 < \ldots < \xi_{m_1}$ are equidistant points given by $\xi_i = \sinh^{-1}(-\frac{K}{c}) + i \cdot \Delta \xi$ ($0 \leq i \leq m_1$) with
  \[ \Delta \xi = \frac{1}{m_1} [\sinh^{-1}(\frac{S-K}{c}) - \sinh^{-1}(-\frac{K}{c})] \]

Analogously, in the $v$-direction we define the following mesh for $0 = v_0 < v_1 < \ldots < v_{m_2} = V$

\[ v_j = d \sinh(\eta_j) \quad (0 \leq j \leq m_2) \]

where:

• $m_2 \geq 1$ is an integer defining the number of points (minus one) in the $v$-axis

• $d > 0$ is a constant that controls the fraction of the mesh points $v_i$ that lie near $v = 0$ (the region we are interested in). In the implementation we used the same value for $c$ proposed in [1]: $d = \frac{V}{500}$

• $\eta_j$ are equidistant points given by $j \cdot \Delta \eta$, where $\Delta \eta = \frac{1}{m_2} \sinh^{-1}(\frac{V}{d})$, for $j = 0, 1, \ldots, m_2$

Both meshes are smooth in the sense that the increments in the mesh are controlled by some constants and $\Delta \xi$ or $\Delta \eta$. We do not delve into the technicalities of the properties of the mesh. These meshes are explored further, for example, in [3]. We took our mesh directly from the paper whose results we aim to emulate.

![Spacial grid for $K = 100$, $m_1 = 50$, $m_2 = 30$, $c = K/5$ and $d = V/500$](image)

### 3.1.2 Derivatives approximation

Given a function $f : \mathbb{R} \to \mathbb{R}$, and the mesh points $x_0 < x_1 < x_2 < \ldots < x_m$, so that $\Delta x_i = x_i - x_{i-1}$ with $i = 1 \ldots m$. The first derivative at a point in the grid, $f'(x_i)$ is approximated using three different schemes, (depending on the region of the grid):

\[
\begin{align*}
f'(x_i) & \approx \alpha_{i,-2} f(x_{i-2}) + \alpha_{i,-1} f(x_{i-1}) + \alpha_{i,0} f(x_i) \quad (10) \\
f'(x_i) & \approx \beta_{i,-1} f(x_{i-1}) + \beta_{i,0} f(x_i) + \beta_{i,1} f(x_{i+1}) \quad (11) \\
f'(x_i) & \approx \gamma_{i,0} f(x_i) + \gamma_{i,1} f(x_{i+1}) + \gamma_{i,2} f(x_{i+2}) \quad (12)
\end{align*}
\]

where:

\[
\begin{align*}
\alpha_{i,-2} & = \frac{\Delta x_i}{\Delta x_{i-1}(\Delta x_{i-1} + \Delta x_i)} , \quad \alpha_{i,-1} = \frac{\Delta x_i - \Delta x_{i-1}}{\Delta x_{i-1} \Delta x_i} , \quad \alpha_{i,0} = \frac{\Delta x_i + 2 \Delta x_{i-1}}{\Delta x_{i-1}(\Delta x_{i-1} + \Delta x_i)} \\
\beta_{i,-1} & = \frac{-\Delta x_{i+1}}{\Delta x_i(\Delta x_i + \Delta x_{i+1})} , \quad \beta_{i,0} = \frac{\Delta x_{i+1} - \Delta x_i}{\Delta x_i \Delta x_{i+1}} , \quad \beta_{i,1} = \frac{\Delta x_i}{\Delta x_{i+1}(\Delta x_{i+1} + \Delta x_i)}
\end{align*}
\]
\[
\gamma_{i,0} = -\frac{2\Delta x_{i+1} - \Delta x_i}{\Delta x_{i+1}(\Delta x_{i+1} + \Delta x_{i+2})}, \quad \gamma_{i,1} = \frac{\Delta x_{i+1} + \Delta x_{i+2}}{\Delta x_{i+1} \Delta x_{i+2}}, \quad \gamma_{i,0} = -\frac{\Delta x_i}{\Delta x_{i+1}(\Delta x_i + \Delta x_{i+1})}
\]

Note that we have backward difference, central difference and forward difference approximations in (11), (11), and (12). The second derivative is approximated by central differences with:

\[
\frac{f''(x_i)}{\Delta x} \approx \frac{3f(x_{i+1}) - 4f(x_i) + f(x_{i-1})}{\Delta x^2}
\]

To approximate the mixed derivative we consider a function of two variables \( f : \mathbb{R}^2 \rightarrow \mathbb{R} \). In the x-direction we use \( x_i \) and \( \Delta x_i \) as in the approximation of the first derivative. Similarly, for the y-direction we take the mesh points \( y_0 < y_1 < y_2 < \ldots < y_n \), so that \( \Delta y_i = y_i - y_{i-1} \) with \( i = 1 \ldots n \). Then, we define the coefficients \( \hat{\beta}_{j,l} \) as for the x-axis but exchanging \( \Delta x \) by \( \Delta y \):

\[
\hat{\beta}_{i,-1} = -\frac{\Delta y_{i+1}}{\Delta y_i(\Delta y_{i+1} + \Delta y_{i+2})}, \quad \hat{\beta}_{j,0} = \frac{\Delta y_{j+1} - \Delta y_j}{\Delta y_j \Delta y_{j+1}}, \quad \hat{\beta}_{j,1} = \frac{\Delta y_j}{\Delta y_{j+1}(\Delta y_j + \Delta y_{j+1})}
\]

We discretize the mixed derivative as:

\[
\frac{\partial^2 f}{\partial x \partial y} (x_i, y_j) \approx \sum_{k,l=-1}^{1} \beta_{i,k} \hat{\beta}_{j,l} f(x_{i+k}, y_{j+l})
\]

These schemes are more complicated than the classical ones we have seen in class because the grid is non-uniform. For example, to derive (11), [5], calling \( u^k_{ij} \) the value of the function at the grid point \( (i,j,k) \) one starts from \( \frac{\partial}{\partial s} (s_i, v_j, t_k) = u^k_{i,j+1} - u^k_{i,j-1} \), if the grid were uniform, \( s_{i+1,j} - s_{i-1,j} = 2 \Delta s \). To approximate (13) one can start from \( \frac{\partial^2 u}{\partial s \partial v} (s_i, v_j, t_k) = u^k_{i,j+1} - u^k_{i,j-1} \) and \( \beta_{j,0} \) by (14). This is obtained applying the usual Taylor expansion around \( u(s + \epsilon^+, v, t) \) and \( u(s - \epsilon^-, v, t) \), with \( \epsilon^+ = s_{i+1} - s_i \), and \( \epsilon^- = s_i - s_{i-1} \). The rest of the derivation is non-trivial and is presented in detail in [3]. Note again that if we had uniform grids \( s_i - s_{i-1} = \delta_s \).

### 3.1.3 Finite Difference Scheme in space

We have formed the following grid in \([0, S] \times [0, V] \):

\[
G = \{(s_i, v_j) : 1 \leq i \leq m_1, \ 0 \leq j \leq m_2 - 1\}
\]

For points inside grid the PDE is approximated by:

- \( \frac{\partial^2 u}{\partial s \partial v} \) and \( \frac{\partial^2 u}{\partial s^2} \) by (13)
- \( \frac{\partial^2 u}{\partial s \partial v} \) by (14)
- \( \frac{\partial u}{\partial s} \) by (11)
- \( \frac{\partial u}{\partial v} \) by (11) if \( v \leq 1 \), by (10) if \( v > 1 \). In the original paper [1] a further conditioned was imposed that the flow in the \( v \)-direction was towards \( v = V \), in order to apply this second approximation.

At the boundary we used the different approximations described below.

### 3.1.4 Implementation of the Boundary Conditions in space

The space domain is restricted to \([0, S] \times [0, V] \) for sufficiently large \( S \) and \( V \). In particular we chose \( S = 8K \) and \( V = 5 \). We specify the conditions of the solution at these boundaries.

The implementation of the boundary condition was one of the main difficulties of the project. Although, they are only briefly described in the paper, their correct implementation is key for the functionality of the numerical scheme. The main difficulty is that at each boundary we need to approximate not only the value of the function, but also the value of its derivatives as this is needed to implement the different ADI schemes. These approximations involve the use of virtual points and of different schemes. Furthermore, the introduction of the boundary conditions \( \frac{\partial u}{\partial \gamma}(S, v, t) = e^{-\gamma t} \) and \( u(S, V, t) = se^{-\gamma t} \) make our boundary conditions time dependent.

\( s = 0 \)
After having discretized the PDE (4) in space we obtain a large system of ODEs of the form:

\[ U'(t) = AU(t) + b(t) \quad (0 \leq t \leq T), \quad U(0) = U_0 \]  

(16)

Where \( A \) is an \((m_1 + 1)(m_2 + 1) \times (m_1 + 1)(m_2 + 1)\) matrix that is obtained through the discretization of the derivatives; \( b(t) \) of dimension \((m_1 + 1)(m_2 + 1)\) is a given vector that depends on the boundary conditions described above, and \( U_0 \) is also an \((m_1 + 1)(m_2 + 1)\)-vector that is obtained through the initial condition.

\( U \) represents, for each \( t > 0 \), the solution to the finite difference scheme at the grid point \( (s, v) \), i.e., the approximation to the exact solution of \( u(s, v, t) \).

To solve this system of ODEs one could use for example the Crank-Nicolson scheme. However, this would involve inverting a block-diagonal matrix, which is computationally intensive. Hence, ADI schemes are preferred.

We start by splitting \( A \) into three matrices. \( A = A_0 + A_1 + A_2 \)

<table>
<thead>
<tr>
<th></th>
<th>( s = 0 )</th>
<th>( s = S )</th>
<th>( v = 0 )</th>
<th>( v = V )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \frac{\partial^2 u}{\partial s^2} )</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial^2 u}{\partial s \partial v} )</td>
<td>0</td>
<td>( s e^{-r f t} )</td>
<td>Central diff. (11)</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial^2 s}{\partial v^2} )</td>
<td>0</td>
<td>Central diff. (13)</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>( \frac{\partial^2 s}{\partial v \partial t} )</td>
<td>0</td>
<td>0</td>
<td>Forward diff. (12)</td>
<td>0</td>
</tr>
<tr>
<td>( u )</td>
<td>Central diff. (13)</td>
<td>Central diff. (13)</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 1: Summary of Boundary Conditions

3.2 ADI schemes

After having discretized the PDE (4) in space we obtain a large system of ODEs of the form:

\[ U'(t) = AU(t) + b(t) \quad (0 \leq t \leq T), \quad U(0) = U_0 \]  

Where \( A \) is an \((m_1 + 1)(m_2 + 1) \times (m_1 + 1)(m_2 + 1)\) matrix that is obtained through the discretization of the derivatives; \( b(t) \) of dimension \((m_1 + 1)(m_2 + 1)\) is a given vector that depends on the boundary conditions described above, and \( U_0 \) is also an \((m_1 + 1)(m_2 + 1)\)-vector that is obtained through the initial condition.

\( U \) represents, for each \( t > 0 \), the solution to the finite difference scheme at the grid point \( (s, v) \), i.e., the approximation to the exact solution of \( u(s, v, t) \).

To solve this system of ODEs one could use for example the Crank-Nicolson scheme. However, this would involve inverting a block-diagonal matrix, which is computationally intensive. Hence, ADI schemes are preferred.

We start by splitting \( A \) into three matrices. \( A = A_0 + A_1 + A_2 \)
The calculation is done in three steps:

1. **Douglas scheme performance.** We start by exploring the Douglas Scheme.

   - **A₀** contains the elements arising from the mixed derivative \( \frac{\partial^2 u}{\partial s \partial v} \). \( A₀ \) corresponds to the discretization of \( \rho \sigma sv \frac{\partial^2 u}{\partial s \partial v} \).

   - **A₁** contains the elements from the derivatives in \( s \), plus half of the contribution of the element in \( u \), i.e., \( A₁ \) corresponds to the discretization of \( \frac{1}{2} \sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + (r_d - r_f) s \frac{\partial u}{\partial s} - \frac{1}{2} r_d u \).

   - **A₂** contains the elements from the derivatives in \( v \), plus half of the contribution of the element in \( u \), i.e., \( A₂ \) corresponds to the discretization of \( \frac{1}{2} \sigma^2 v^2 \frac{\partial^2 u}{\partial v^2} \).

A similar decomposition is performed for \( b(t) \): \( b(t) = b₀(t) + b₁(t) + b₂(t) \). Finally, the following functions are defined:

\[
F_j(t, X) = A_j X + b_j(t), \quad X \in \mathbb{R}^m
\]

These equations were very difficult to implement because of their dimensionality and because different elements of the matrix \( A \) are approximated using different schemes. Thus, its implementation is error prone. However, since these matrices have a high number of null entries, a simple visual inspection can help in spotting potential mistakes. To help with this, we implemented some auxiliary functions (\texttt{check\_A()} and \texttt{check\_A\_b()}) to visualize the non-zero parts of the matrices. Figure 2 shows the non-zero parts of the \( A \) matrices, we obtained once our implementation was finally correct.

![Figure 2: Visualization of A matrices. A₀, A₁, A₂](image)

In'T Hout and Foulon propose four different ADI schemes to solve this system of ODEs. Each of these schemes is based on calculating iteratively \( U^n(t) \) from \( U^{n-1}(t) \), and starting from the initial conditions in \( U^0(t) \), \( n = 1, 2, \ldots \), with \( t = n \times \Delta t \) at each time step. In the case of the European call option the initial condition is given by (5), so we take: \( u_{i,j}(t = 0) = max(0, s_i - K) \) for \( 0 \leq i \leq m_1 \) and \( 0 \leq j \leq m_2 \). We do not need to approximate the derivatives at \( t = 0 \), since they are not needed to approximate \( u_{i,j}(0) \).

We started from the simplest Douglas Scheme and later implemented all four schemes to compare their performance. We start by exploring the Douglas Scheme.

### 3.2.1 Douglas scheme

The calculation is done in three steps:

1. \( Y_0 = U_{n-1} + \Delta t F(t_{n-1}, U_{n-1}) \). Hence, \( Y_0 = (I + \Delta t A) U_{n-1} + b(t_{n-1}) \)

2. \( Y_j = Y_{j-1} + \theta \Delta t [F_j(t_{n}, Y_j) - F_j(t_{n-1}, U_{n-1})] \).

Hence, \( Y_j = (I - \theta \Delta t A)_{j-1} ^{-1} (Y_{j-1} - \theta \Delta t A_j U_{n-1}) + (I - \theta \Delta t A_j)^{-1} (\theta \Delta t) (b_j(t_n) - b_j(t_{n-1})) \quad j = 1, 2 \)

3. \( U_n = Y_2 \)

Remarks:

- Since the boundary conditions are time dependent, the term \( b_j(t_n) - b_j(t_{n-1}) \) does not vanish.

- The parameter \( \theta \in \mathbb{R} \) has the same meaning as studied in class for the \( \theta \)-schemes. For example if \( \theta = 0 \) we get a fully explicit scheme. Indeed, for example, it is easy to see that for \( \theta = 0 \): \( U_n = Y_2 = Y_1 = Y_0 = (I + \Delta t A) U_{n-1} + b(t_{n-1}) \).
The main difficulty lies in building the A matrix. Once this matrix is built, it is just a matter of implementing a short iteration.

The order of accuracy of the Douglas scheme for any value of $\theta$ is 1 [1]. Hence, the authors explore three other schemes with better order of accuracy.

3.2.2 Advanced schemes
The other three schemes are the Craig-Sneyd (CS) scheme, the Modified Craig-Sneyd (MCS) scheme, and the Hundsdorfer-Verwer (HV) scheme. These schemes have an order of accuracy of 2 for given values of $\theta$.

Here we only specify in detail the more general one, the MCS scheme, for brevity. The other two can be found in [1].

**Modified Craig-Sneyd**
The calculation is done in 6 steps:

1. $Y_0 = U_{n-1} + \Delta t F(t_{n-1}, U_{n-1})$
2. $Y_j = Y_{j-1} + \theta \Delta t (F_j (t_n, Y_j) - F_j (t_{n-1}, U_{n-1}))$ (j = 1, 2)
3. $\bar{Y}_0 = Y_0 + \theta \Delta t (F_0 (t_n, Y_2) - F_0 (t_{n-1}, U_{n-1}))$
4. $\bar{Y}_0 = \bar{Y}_0 + \left( \frac{1}{2} - \theta \right) \Delta t (F (t_n, Y_2) - F (t_{n-1}, U_{n-1}))$
5. $\bar{Y}_j = \bar{Y}_{j-1} + \theta \Delta t \left( F_j (t_n, \bar{Y}_j) - F_j (t_{n-1}, U_{n-1}) \right)$ (j = 1, 2)
6. $U_n = \bar{Y}_2$

The first two steps are the same in all four schemes. After the first two steps the Douglas scheme jumps to step 6. The CS scheme merges steps 3 and 4 in a single step, it does not consider the intermediate point $\bar{Y}_0$. The HV scheme does not consider the intermediate point $\bar{Y}_0$ either; it takes $\theta = 0$ in step 4; and has a slightly different expression for step 5.

4 Implementation in Python
In this section we describe briefly the structure of the implementation of the schemes in Python. The structure of the implementation follows a similar pattern as the structure of the paper.

- We start the implementation by generating the mesh grid in 3.1.1, using a `getGrid()` function. With this grid, we are able to initialize the $u$ matrix based on (5).
- Since different schemes are involved in the approximation of derivatives, we designed functions like `cAlpha()`, `cBeta()` to generate parameters according to section 3.1.2.
- These parameters are then filled into the A matrices in loops.
• For special cases such as when \( s = S \), the value of the derivative is filled into vector \( b \) straight away.

• The \( fdm\_heston() \) function is designed to implement the ADI scheme of choice to generate the final \( u \) matrix. For example, it follows the process detailed in 3.2.1 when implementing the DO scheme.

• For points which are not on the grid, we interpolate to get a value.

• Finally, the \( getError() \) function calculates the percentage error when compared with the Heston pricer based on Fourier Transform.

5 Numerical results

First of all, we did a sanity check making sure that higher volatilities and higher initial stock prices lead to a higher call option price. We plotted the call option price for implausible values of \( S \) and \( V \). But this simple check allowed us to spot numerous bugs in the implementation.

![Figure 4: European call price for different values of V and S](image)

We performed several analysis to test the properties of the different schemes. For the sake of comparison we used the same scenarios as in the original paper. In all scenarios we took \( \theta = 0.8 \), since this value falls within the unconditional stability region of all the implemented schemes.

<table>
<thead>
<tr>
<th>Case 1</th>
<th>Case 2</th>
<th>Case 3</th>
<th>Case 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa )</td>
<td>1.5</td>
<td>3</td>
<td>0.6067</td>
</tr>
<tr>
<td>( \eta )</td>
<td>0.04</td>
<td>0.12</td>
<td>0.0707</td>
</tr>
<tr>
<td>( \sigma )</td>
<td>0.3</td>
<td>0.04</td>
<td>0.2928</td>
</tr>
<tr>
<td>( \rho )</td>
<td>-0.9</td>
<td>0.6</td>
<td>-0.7571</td>
</tr>
<tr>
<td>( r_d )</td>
<td>0.025</td>
<td>0.01</td>
<td>0.03</td>
</tr>
<tr>
<td>( r_f )</td>
<td>0</td>
<td>0.04</td>
<td>0</td>
</tr>
<tr>
<td>( T )</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>( K )</td>
<td>100</td>
<td>100</td>
<td>100</td>
</tr>
</tbody>
</table>

\( m_1 = 50, m_2 = 25, m_3 = 20, V = 5, c = K/5, d = V/10, S_0 = 100, v_0 = 0.04, \theta = 0.8 \)

Table 2: Parameters for the Heston model and European call options under different scenarios

Under each of these scenarios we compared the price obtained through each scheme with the price obtained through the Fourier Transform (FT) approach, which we took as the reference. The pricing through FT is based on the Zanadu Notebook “Heston Pricing” by Antoine Jacquier, expanded to incorporate \( r_f \). The % error was defined as:

\[
Error = \frac{\text{price}_\text{scheme} - \text{price}_\text{FT}}{\text{price}_\text{FT}} \times 100
\]

We obtained the following results:
Table 3: European Call Option prices under different ADI schemes

We notice that even though the more involved schemes have a better order of accuracy, the final error is, in general, not smaller than that of the Douglas scheme. The biggest errors are found in case 3, which might be due to the fact that under this scenario the Feller condition is only just met.

Next, we analyze how the error changes when as the number of nodes in the $v$–axis, $m_2$ varies. For this part we took $m_1 = 2 m_2$. Also, motivated by the theoretical stability and accuracy results discussed in [1], we assigned as the paper the following values to $\theta$ in each of the schemes: Do and CS, $\theta = \frac{1}{2}$; MCS, $\theta = \frac{1}{4}$; and HV, $\theta = 1 - \frac{1}{2} \sqrt{2}$.

In this part we consider the absolute value of the error.

Figure 5: Pricing error under different volatility step sizes
We observe that under all scenarios the error decreases as the spatial grid becomes finer.

Finally, we analyzed how the error changes as the time step size varies. In most cases we see that as the time step size decreases the error also decreases. This is not true in the case of scenario 3, and it might be related to the stability of the scheme when the Feller condition is just met.

![Figure 6: Pricing error under different time step size](image)

We observe that the error changes in pretty much the same way in all schemes as we change $m_2$ or $m_3$. This is in line with the finding in [1] that the errors in all four schemes are very correlated.

All in all, we see that the accuracy of these methods for a small number of grid points is high. However, to complete the analysis we would have to implement a classical Crank-Nicolson scheme to solve the same problem and compare accuracy and speed of computation.

References

[1] K.J. In ’t Hout, S. Foulon, ADI finite difference schemes for option pricing in the Heston model with correlation.


