On the study of Vanilla and Barrier Options under a local volatility model using finite difference methods

Bassam SINAN

15 March 2022

Contents

1 Introduction

We consider a local volatility model in which the stock price has the dynamics of:

$$\frac{dS_t}{S_t} = r\,dt + \sigma(t, S_t)\,dB_t \tag{1}$$

under the risk neutral measure. Here r is the constant riskfree rate, B is a Brownian motion and $\sigma(.)$ is a deterministic function which reflects the instantaneous volatility of the stock return at time t when the stock price is at level s.

For convenience, we will consider the following explicit form of the local volatility function:

$$\sigma(t, s) = \left(1 + \frac{t}{30}\right)\left[0.1 + 0.4exp\left(-\frac{s}{50}\right)\right] \tag{2}$$

(Remark: the specification in (2) is a totally made-up toy example. The industrial practice is to construct the local volatility function from market prices of call/put options using the so-called Dupire's formula)

If we want to price a simple European option of maturity of T with payoff function g(S_T), then using Feynman-Kac formula the option value function V(t,s) will satisfy the PDE:

$$\frac{\partial V}{\partial t} + rs\frac{\partial V}{\partial s} + \frac{\sigma^2(t,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} - rV = 0, \qquad t < T \tag{3}$$

$$V(T, s) = g(s), \qquad t = T \tag{4}$$

2 European Option

2.1 Reformulating the PDE as an initial condition problem

We want to reformulate the PDE as an initial condition problem. Let $\tau = T - t$. Then we can rewrite:

$$\begin{aligned}\frac{\partial V}{\partial t} &= \frac{\partial V}{\partial \tau}\frac{\partial \tau}{\partial t} \\ &= -\frac{\partial V}{\partial \tau}\end{aligned}$$

$$\sigma(\tau, s) = \left(1 + \frac{\tau}{30}\right)\left[0.1 + 0.4exp\left(-\frac{s}{50}\right)\right]$$

Hence, (3) and (4) become:

$$\frac{\partial V}{\partial \tau} = rs\frac{\partial V}{\partial s} + \frac{\sigma^2(\tau,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} - rV = 0, \qquad \tau > 0 \tag{5}$$

$$V(\tau, s) = g(s), \qquad \tau = 0 \tag{6}$$

2.2 Explicit, fully implicit and Crank-Nicolson scheme to solve the PDE

In the next sections, to simplify notations we will replace t with τ so we will write the initial condition as t=0 (instead of τ=0) and so on and so forth.

Boundary condition

In this example we will work with the domain of (t,s) directly which is truncated to $D := [0,T] \times [s_{min}, s_{max}]$. Since a call option is involved, we impose the boundary conditions:

$$V(t,s_{min}) = \ell(t,s_{min}) = 0, \qquad V(t,s_{max}) = u(t,s_{max}) = s_{max} - Ke^{-rt} \qquad \text{for all } t < T \tag{7}$$

Grid specification

We construct a uniform grid over D with $N+1$ points along the time dimension and $M+1$ points along the space dimension. Let

$$\triangle t := \frac{T}{N}, \qquad \triangle x := \frac{s_{max} - s_{min}}{M}.$$

Then the values of the grid points are given by

$$\begin{aligned} t_n &:= n\triangle t, && n = 0,1,...,N \\ s_k &:= s_{min} + k\triangle x, && k = 0,1,...,M \end{aligned}$$

2.2.1 The Explicit Scheme

Under explicit scheme the PDE can be discretised as

$$\begin{aligned} \frac{V_k^{n+1} - V_k^n}{\triangle t} &= \frac{\sigma^2(t,s)s_k^2}{2}\frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + rs_k\frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - rV_k^n \\ \implies \quad V_k^{n+1} &= \left(\frac{\triangle t}{\triangle x^2}\frac{\sigma^2(t,s)s_k^2}{2} - \frac{\triangle t}{2\triangle x}rs_k\right)V_{k-1}^n + \left(1 - \frac{\triangle t}{\triangle x^2}\sigma^2(t,s)s_k^2 - \triangle tr\right)V_k^n \\ &\quad + \left(\frac{\triangle t}{\triangle x^2}\frac{\sigma^2(t,s)s_k^2}{2} + \frac{\triangle t}{2\triangle x}rs_k\right)V_{k+1}^n \\ &= A_kV_{k-1}^n + (1+B_k)V_k^n + C_kV_{k+1}^n \end{aligned}$$

for $k = 1,2,...,M-1$, where

$$A_k := \frac{\triangle t}{\triangle x^2}\frac{\sigma^2(t,s)s_k^2}{2} - \frac{\triangle t}{2\triangle x}rs_k, \qquad B_k := -\frac{\triangle t}{\triangle x^2}\sigma^2(t,s)s_k^2 - \triangle tr, \qquad C_k := \frac{\triangle t}{\triangle x^2}\frac{\sigma^2(t,s)s_k^2}{2} + \frac{\triangle t}{2\triangle x}rs_k \tag{8}$$

Recall that the $M-1$ recursive equations can be summarised using matrix notation

$$\underbrace{\begin{bmatrix} V_0^{n+1} \\ V_1^{n+1} \\ V_2^{n+1} \\ \vdots \\ \vdots \\ V_{M-1}^{n+1} \\ V_M^{n+1} \end{bmatrix}}_{=:V^{n+1}} = \underbrace{\begin{bmatrix} 1 & 0 & 0 & \cdots & & & & 0 & 0 \\ A_1^n & 1+B_1^n & C_1^n & 0 & \cdots & & & 0 & 0 \\ 0 & A_2^n & 1+B_2^n & C_2^n & 0 & \cdots & & & \\ & & & & \ddots & & & & \\ \vdots & & & & & A_{M-2}^n & 1+B_{M-2}^n & C_{M-2}^n & 0 \\ & & & & & 0 & A_{M-1}^n & 1+B_{M-1}^n & C_{M-1}^n \\ 0 & 0 & \cdots & & & 0 & 0 & 0 & 1 \end{bmatrix}}_{=:\mathbb{I}+L^n} \underbrace{\begin{bmatrix} V_0^n \\ V_1^n \\ V_2^n \\ \vdots \\ \vdots \\ V_{M-1}^n \\ V_M^n \end{bmatrix}}_{=:V^n}$$

Here $\mathbb{I}$ is an $(M+1)\times(M+1)$ identity matrix and L^n is an $(M+1)\times(M+1)$ matrix in form of

$$L^n := \begin{bmatrix} 0 & 0 & 0 & \cdots & & & & 0 & 0 \\ A_1^n & B_1^n & C_1^n & 0 & \cdots & & & 0 & 0 \\ 0 & A_2^n & B_2^n & C_2^n & 0 & \cdots & & & \\ & & & & \ddots & & & & \\ \vdots & & & & & A_{M-2}^n & B_{M-2}^n & C_{M-2}^n & 0 \\ & & & & & 0 & A_{M-1}^n & B_{M-1}^n & C_{M-1}^n \\ 0 & 0 & \cdots & & & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

In our specific example, A, B and C (and in turn the matrix L) have no dependence on n and hence can be pre-set before running the loop.

Recall that the system $V^{n+1} = (\mathbb{I} + L^n)V^n$ doesn't give us the correct value of V_0^{n+1} and V_M^{n+1}. We need to manually overwrite the first and last entry of the vector to incorporate the boundary conditions at $s = s_0 = s_{min}$ and $s = s_M = s_{max}$. The complete recursive algorithm is:

$$V^{n+1} = B^{n+1}((\mathbb{I} + L^n)V^n) \tag{10}$$

where $B^{n+1}(\cdot)$ is an operator which overwrites the first and last entry of the input vector to $\ell_0^{n+1} := \ell(t_{n+1}, x_0)$ and $u_M^{n+1} := u(t_{n+1}, x_M)$.

2.2.2 The Fully Implicit Scheme

Under fully implicit scheme the PDE is now discretised as

$$\begin{aligned} \frac{V_k^n - V_k^{n-1}}{\triangle t} &= \frac{\sigma^2(t,s)s_k^2}{2}\frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + r s_k \frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - r V_k^n \\ \implies \quad V_k^{n-1} &= -\left(\frac{\triangle t}{\triangle x^2}\frac{\sigma^2(t,s)s_k^2}{2} - \frac{\triangle t}{2\triangle x} r s_k\right) V_{k-1}^n + \left(1 + \frac{\triangle t}{\triangle x^2}\sigma^2(t,s)s_k^2 + \triangle t r\right) V_k^n \\ &\quad - \left(\frac{\triangle t}{\triangle x^2}\frac{\sigma^2(t,s)s_k^2}{2} + \frac{\triangle t}{2\triangle x} r s_k\right) V_{k+1}^n \\ &= -A_k V_{k-1}^n + (1 - B_k)V_k^n - C_k V_{k+1}^n \end{aligned}$$

for A, B and C are defined previously. After taking boundary condition into account, a matrix representation is

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & & & & 0 & 0 \\ -A_1^n & 1-B_1^n & -C_1^n & 0 & \cdots & & & 0 & 0 \\ 0 & -A_2^n & 1-B_2^n & -C_2^n & 0 & \cdots & & & \\ & & & & \ddots & & & & \\ \vdots & & & & & -A_k^n & 1-B_k^n & -C_k^n & 0 \\ & & & & & 0 & -A_{M-1}^n & 1-B_{M-1}^n & -C_{M-1}^n \\ 0 & 0 & \cdots & & & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} V_0^n \\ V_1^n \\ V_2^n \\ \vdots \\ \vdots \\ V_{M-1}^n \\ V_M^n \end{bmatrix} = \begin{bmatrix} \ell_0^n \\ V_1^{n-1} \\ V_2^{n-1} \\ \vdots \\ \vdots \\ V_{M-1}^{n-1} \\ u_M^n \end{bmatrix}$$

In matrix form, it can be represented as

$$(\mathbb{I} - L^n)V^n = B^n(V^{n-1}). \tag{11}$$

To solve V^n, we implement the Thomas algorithm which solves system of equation $Ax = d$ with A being a tridiagonal

matrix of the form

$$\begin{bmatrix} b_0 & c_0 & 0 & \cdots & & & & 0 & 0 \\ a_1 & b_1 & c_1 & 0 & \cdots & & & 0 & 0 \\ 0 & a_2 & b_2 & c_2 & 0 & \cdots & & & \\ & & & & \ddots & & & & \vdots \\ \vdots & & & & & a_{M-2} & b_{M-2} & c_{M-2} & 0 \\ & & & & & 0 & a_{M-1} & b_{M-1} & c_{M-1} \\ 0 & 0 & \cdots & & & 0 & 0 & a_M & b_M \end{bmatrix}$$

2.2.3 The Crank-Nicolson Scheme

$$\frac{V_k^n - V_k^{n-1}}{\triangle t} = \theta\left(\frac{\sigma^2(t,s)s_k^2}{2}\frac{V_{k+1}^n - 2V_k^n + V_{k-1}^n}{\triangle x^2} + rs_k\frac{V_{k+1}^n - V_{k-1}^n}{2\triangle x} - rV_k^n\right) + (1-\theta)\left(\frac{\sigma^2(t-1,s)s_k^2}{2}\frac{V_{k+1}^{n-1} - 2V_k^{n-1} + V_{k-1}^{n-1}}{\triangle x^2} + rs_k\frac{V_{k+1}^{n-1} - V_{k-1}^{n-1}}{2\triangle x} - rV_k^{n-1}\right)$$

Based on the same idea as in the analysis of the explicit/implicit scheme, the recursive equations and the boundary conditions can be summarised by the following matrix notation:

$$(\mathbb{I} - \theta L^n)V^n = B^n(\mathbb{I} + (1-\theta)L^{n-1})V^{n-1}). \tag{12}$$

2.3 Prices obtained with the three schemes

As we can see on the Table below, we observe little difference between the three schemes with prices being equal up to 2 d.p. We also find these prices converge towards the BS price (see jupyter notebook).

Stock Price	Explicit Scheme Prices	Implicit Scheme Prices	Crank-Nicolson Scheme Prices
80.0	0.708623	0.709268	0.708946
85.0	1.399338	1.399475	1.399406
90.0	2.534719	2.534094	2.534406
95.0	4.237428	4.236102	4.236765
100.0	6.583896	6.582263	6.583079
105.0	9.581941	9.580521	9.581231
110.0	13.170821	13.169979	13.170400
115.0	17.242413	17.242219	17.242316
120.0	21.671686	21.671972	21.671829

Figure 1: Table of prices obtained for a European Call with K=100, r=1%,d=0,T=1year

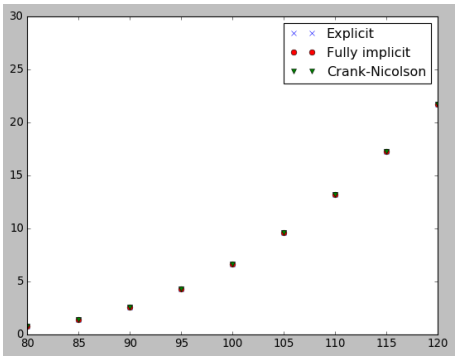

Figure 2: Plot of prices obtained for a European Call with K=100, r=1%,d=0,T=1year

3 Implied Volatility

Prices (namely under the Black Scholes model) use volatility as a parameter. This volatility is referred to as implied volatility. In particular, using the prices available in the market[1], we can extract an implied volatility under the BS model.[2]

[2]Prices are usually quoted in terms of implied volatility in the market
[2]Prices correspond to one and only one volatility parameter

While realized volatility looks at past returns and hence is also called the historical volatility, implied volatility looks towards the future as it is implied from prices with a maturity in the future.

Because the prices of any financial instruments are also sensitive to the liquidity available in the market, the implied volatility reflect the market consensus on the forward volatility of the asset.

In the previous section, we have computed prices for a European Call option under the three schemes we have studied. We now focus on the Explicit Scheme. We fix the initial Stock Price at S_0=100 and compute the European Call Prices over a range of strikes $K \in [70, 130]$ under the local volatility model specified in 2.

For each price we compute the implied volatility and plot it against K.

3.0.1 Local volatility model

As we can see on the two figures below, our local volatility model is downward sloping. This seems consistent with the implied volatility slope we have derived from our prices:

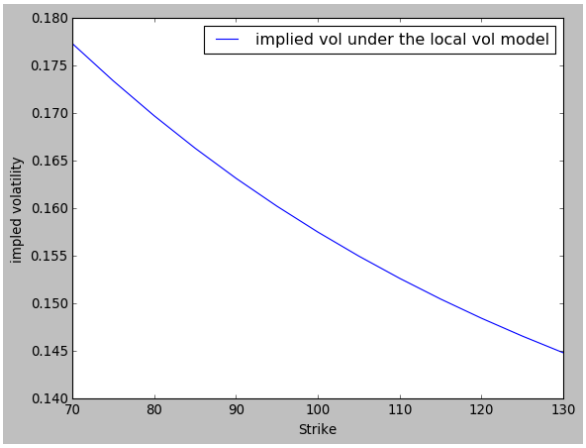

Figure 3: Implied vol extracted from our prices under the local volatility model we have specified in (2)

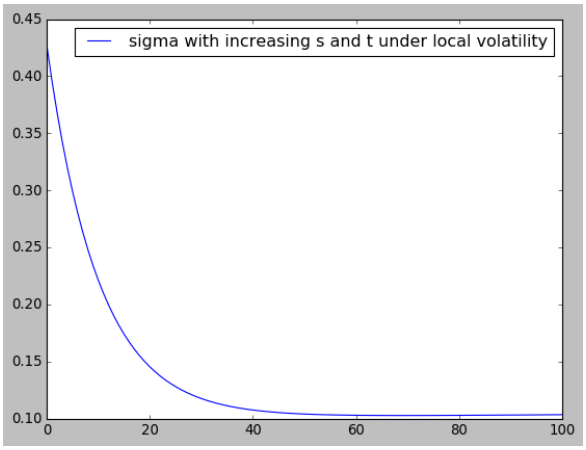

Figure 4: Plotting sigma(t,s) under (2) with increasing s and t

The reason why we would want to have a downward slope local volatility is to fit the implied volatility slope observed in the market.

This phenomenon is referred in the literature as skew. Indeed, we observe in a downward slope across strikes called skew (example: equity markets). In other asset classes such as FX, this shape is a smile.

This skew indicates that options with a lower strike display higher levels of implied volatility. There are several reasons behind but are beyond the scope of this project.

Since vanilla options (specifically in our example a call) are long vega, options with lower strike should be hence more expensive (ceteris paribus) because of the presence of skew. This is emphasized by the table below.

Bin_range	DIP_prices_constant_volatility	DIP_prices_local_volatility	% Difference
60	0.049897	0.256776	4.146117
70	0.713040	1.232085	0.727933
80	3.106387	3.495763	0.125347
90	5.429516	5.479137	0.009139

Figure 5: Prices across strikes under the local volatility model for S=100, r=1%, T=1year

3.0.2 Constant volatility model

As we would have expected, the implied volatility is constant across strikes:

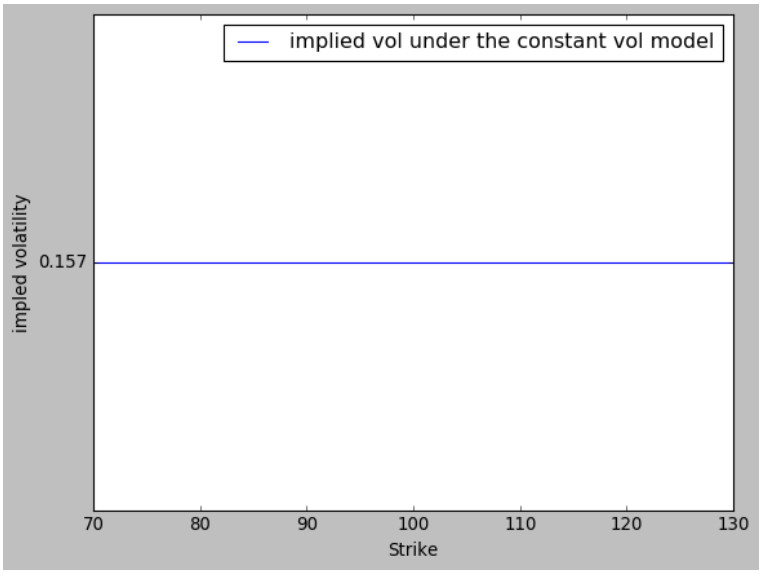

Figure 6: Implied vol extracted from our prices under the constant volatility model (σ =15.7%) under the explicit scheme

4 Down-and-in European Put

The Down-and-in Put (DIP) Payoff is given by:

$$(K - S_T)^+ \mathbb{1}_{(L_T \leqslant B_{in})]} \tag{13}$$

where $L_t := \inf_{0 \leqslant u \leqslant t} S_u$ is the running minimum of the stock price up to time t, and B_{in} is the knock-in level of the option.

We want to derive a general approach for pricing the DIP. It may not be clear upfront how to write down the precise boundary condition because we may not know the European call option price function (especially if we do not work under the Black-Scholes model). A more convenient approach is to utilise the "in-out parity" i.e Vanilla Put = Down-and-in Put + Down-and-out Put.

It is straightforward to formulate the PDE satisfied by a knock-out barrier option. All we need to do is to introduce a suitable boundary condition where the option value is set to zero at the knock-out level.

Let V(t,s) be the time-t value of a down-and-out barrier put option (DOP) with knock-out level B_{ko} (< S0) when the current stock price is s.

For any t, we expect $V(t, B_{ko})$ = 0 because the current stock price has now crossed the barrier and the option vanishes.

Under the Black-Scholes model, V satisfies the PDE:

$$\frac{\partial V}{\partial t} = rs\frac{\partial V}{\partial s} + \frac{\sigma^2(t,s)s^2}{2}\frac{\partial^2 V}{\partial s^2} - rV = 0, \qquad t > 0 \tag{14}$$

$$V(T, B_{ko}) = 0, \qquad t > 0 \tag{15}$$

$$V(T, s) = (K - s)^+ \mathbb{1}_{(s \geqslant B_{ko})]}, \qquad t = 0 \tag{16}$$

We are interested in the value of V(t,s) over (t,s)$\in$ $[0, T]\times[B_{ko}, s_{max}]$. We already have the boundary condition V(t,B_{ko}) = 0. Along s = s_{max} , we expect V(t, s_{max}) = 0 as well since the put option will be deep OTM and its value will tend to 0.

We derive the Explicit and Fully Implicit Scheme using the exact same approach as in section 2 (see code for further detail) and we get the DIP Price by substracting the European Vanilla Put Price to the DOP Price we have computed. We provide the results below using the implicit scheme. We find that the explicit scheme tends to be unstable when handling edge cases such as high volatilities.

We fix the initial stock price as S0 = 100, strike as K = 100 and maturity as T = 1 year. Below are down-and-in barrier put option prices over different knock-in levels $B_{in} \in \{60, 70, 80, 90\}$ under both the local volatility model and a constant volatility model with (t,s) = 15.7% for all t, s.

Bin_range	DIP_prices_constant_volatility	DIP_prices_local_volatility	% Difference
60	0.049897	0.256776	4.146117
70	0.713040	1.232085	0.727933
80	3.106387	3.495763	0.125347
90	5.429516	5.479137	0.009139

Figure 7: DIP Prices under the implicit scheme over a range of KI Barriers

We can make two observations from the prices derived in the table above:

(i) DIP are more expensive under the local volatility model

(ii) The % difference fades away as B_{in} gets closer to K

(i) A DIP can be broken down into two components. A vanilla put and a digital put. The digital put is long skew (refer to `https://bookdown.org/maxime_debellefroid/MyBook/barrier-options.html` for example for further details about the skew sensitivity) hence the DIP is long skew which explains why this DIP option costs more under the local volatility model.

(ii) DIP is short the Barrier. Indeed, the higher the barrier (i.e the closer it gets to the price) the more likely the DIP is to KI and hence the higher the price is. As the barrier gets closer to the stock price, the DIP price converges to the vanilla put price. Hence, the skew effect has less impact on the DIP. We can also see it like this. When replicating the DIP, the notional needed to replicate the digital leg gets smaller and smaller as the barrier goes up which means that the skew effect has less impact.

5 Code Link

https://github.com/BassamSINAN/Finite-Difference-Methods-for-Vanilla-and-Barrier-Options

References

[1] Cristopher Salvi (2022), *Numerical Methods, course taught as part of the modules of the MSc in Mathematics and Finance at Imperial College*

[2] Adel Osseiran and Mohamed Bouzoubaa (2010) *Exotic Options and Hybrids: A Guide to Structuring, Pricing and Trading*

[3] Maxime Debellefroid *The Derivatives Academy*
Available at: https://bookdown.org/maxime_debellefroid/MyBook/