Finite difference pricing and hedging of European options

A model problem: Black-Scholes European put

Model specification and preliminaries

The classical model for discussing the finite difference pricing of European options is the Black-Scholes model. It is a good test case not because it is a realistic model in practice, but because it is *the* building block for more adequate models, it has many of the characteristic features of option pricing equations, with well-studied poperties and known closed-form solutions to compare the numerical solution against.

In the Black-Scholes model, the time t evolution of the risk-free value $V(S,t)$ of a European option on a stock with price S, follows the Black-Scholes PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0, \quad S \in [0,\infty), t \in (0,T). \tag{6.1}$$

The option expires at some time $T > 0$. The model parameters σ and r are the volatility and prevailing interest rate. They are assumed constant and are non-negative. A typical set of parameters would be in the range $\sigma = 0.1, \ldots, 0.6$, $r = 0.01, \ldots, 0.08$, $T = 0.1, \ldots, 10$. The task is to find $V(S_0, 0)$, today's option value, where S_0 is the current value of the stock.

To do this, the PDE (6.1) is solved *backwards* in time, from a terminal condition at the expiry T of the contract, which is given by the payoff, say G, of the European option,

$$V(S,T) = G(S), \quad S \in [0,\infty). \tag{6.2}$$

The Black-Scholes equation (6.1) is the *Feynman-Kac* PDE to the stochastic process

$$\mathrm{d}S_t = rS_t\,\mathrm{d}t + \sigma S_t\,\mathrm{d}W_t, \tag{6.3}$$

which is of course geometric Brownian motion. Conversely, $V(S_t,t)$ is the discounted expectation of the payoff,

$$V(S_t,t) = \mathrm{e}^{-r(T-t)}\mathbb{E}(G(S_T)|S_t),$$

given the current stock price is S_t and where S follows (6.3).

For a European put option, for instance, the terminal condition (payoff) at expiry T is

$$V(S,T) = \max(K-S,0) \quad S \in [0,\infty), \tag{6.4}$$

where $K > 0$ is the exercise or strike price.

The Black-Scholes model is special in that it assumes, among other things, known constant volatility (the relative variance of returns). This makes it analytically solvable such that a new numerical scheme can be tested against the closed-from solutions before applying it to a more complex case with unknown solution. The Black-Scholes value of a European put is

$$P(S,t) = K\mathrm{e}^{-r(T-t)}N(-d_2) - SN(-d_1), \tag{6.5}$$

where

$$d_1 = \frac{\ln(S/K) + (r+\sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}, \tag{6.6}$$

$$d_2 = \frac{\ln(S/K) + (r-\sigma^2/2)(T-t)}{\sigma\sqrt{T-t}}. \tag{6.7}$$

Aside its specific form and closed-form solution, the Black-Scholes model is representative for the general structure of pricing equations. When discussing numerical schemes for the Black-Scholes model, we will have in the back of our minds the slightly more general PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2(S,t)\frac{\partial^2 V}{\partial S^2} + \mu(S,t)\frac{\partial V}{\partial S} - r(S,t)V = 0, \tag{6.8}$$

which encompasses a wider range of models by suitable choice of the functions σ, μ, and r. By comparison, in the Black-Scholes model

$$\sigma(S,t) = \sigma S, \quad \mu(S,t) = rS, \quad r(S,t) = r,$$

with non-negative constants σ and r.

A key feature is that the coefficients $\sigma(S,t)$ and $\mu(S,t)$ degenerate for $S \to 0$, which is related to the process for S being bounded below by zero for positive starting value. This means $S = 0$ is a "natural" boundary for the PDE. At $S = 0$, the PDE reduces to the ODE

$$\frac{\partial V}{\partial t}(0,t) - rV(0,t) = 0, \tag{6.9}$$

$$V(0,T) = G(0). \tag{6.10}$$

This ODE can be solved independent of the PDE, with solution

$$V(0,t) = \mathrm{e}^{-r(T-t)}G(0).$$

No additional conditions for the PDE can (have to) be imposed at $S = 0$. We will call this a *natural boundary condition.*

We also need to demand an asymptotic condition for $S \to \infty$. Looking towards the computation, we restrict the positive axis to a finite interval $[0, S_{max}]$, and therefore need to set a boundary condition at $S = S_{max}$. If S_{max} is sufficiently large, the impact of this boundary value on the solution in the region of interest will be small. This becomes clear if we think of the boundary value as a payoff which is realised when S hits (the boundary point)

S_{max}. For large S_{max}, this event becomes unlikely and has little impact on the expected payoff, i.e. the solution $V(S,t)$, for relevant S.

For computational efficiency, however, S_{max} should not be unnecessarily large and one aims to restrict the computation to as small a region as possible. It then becomes a relevant question how to approximate the solution at S_{max} most accurately. If we could set the exact value of the solution to the unrestricted problem on $(0,\infty)$ as boundary condition at S_{max}, the solution to the IBVP would coincide with the former, but finding this solution is exactly the task of this computation. It is often possible to find an asymptotically accurate approximation to the solution, i.e. one that becomes more accurate for growing S, without solving the full PDE. The impact of this approximation on the overall solution is bounded by the error at the boundary, from the maximum principle for parabolic PDEs. In practice, the error in the region of interest is usually much smaller.

Take the example of the put. The put is unlikely to get *in-the-money* by expiry, if S is "large" today, such that the value $V(S,t) \to 0$ for $S \to \infty$. An asymptotic approximation is therefore $V(S_{max},t) = 0$. We focus on this example here and consider the implementation of different payoffs in Section 6.4.

A reasonable choice for "large" S_{max} could be guided by the current spot value plus a sufficiently large number of standard deviations. The variance scales with the expiry T. With common values of the volatility in the range of at most 50% p.a. (per year), a interest rate of around 5% and expiry in less than 20 years, picking $S_{max} = S_0 \exp(3\sqrt{T})$ should be good enough for practical applications. Then

$$\begin{aligned} \mathbb{P}(S_t > S_{max}) = \mathbb{P}(\log S_t > \log S_{max}) &= \mathbb{P}(\sigma W_t > \log(S_{max}/S_0) - (r - \tfrac{1}{2}\sigma^2)t) && (6.11)\\ &\leq \Phi(-5) && (6.12)\\ &\approx 2.87 \cdot 10^{-7}. && (6.13) \end{aligned}$$

A more refined calculation would use hitting probabilities via the running maximum. We will estimate the approximation error in detail in Section 6.4.

The resulting initial-boundary value problem is

$$\begin{aligned} \frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0 \qquad & S \in [0, S_{max}), t \in (0,T), && (6.14)\\ V(S_{max},t) = 0 \qquad & t \in (0,T], && (6.15)\\ V(S,T) = \max(K-S,0) \qquad & S \in [0, S_{max}]. && (6.16) \end{aligned}$$

It is interesting to note as an aside that the Black-Scholes PDE is invariant under rescaling of the S coordinate by a constant factor.

Remark 6.1.1 (Price vs. log-price)**.** *Due to the log-normality of the underlying distribution in the Black-Scholes model, the transformation $X = \log S$, $U(x,t) = V(S,t)$, leads to the constant coefficient PDE*

$$\frac{\partial U}{\partial t} + \frac{1}{2}\sigma^2 \frac{\partial^2 U}{\partial X^2} + (r - \sigma^2/2)\frac{\partial U}{\partial X} - rU = 0. \tag{6.17}$$

This is the Feynman-Kac PDE for a Brownian motion with variance σ^2 and drift $\mu = r - \sigma^2/2$. Further transformations, namely

1. *removing the drift via $x = X - \mu t$;*

2. *reverting the direction of time to measure* time-to-maturity*, and rescaling for unit variance per unit time,* $\tau = \sigma^2(T-t)$*;*

3. *inflating (i.e., undoing discounting of) the price,* $u = \exp(r(T-t))U$*,*

lead to the (forward) heat equation

$$\frac{\partial u}{\partial \tau} = \frac{1}{2}\frac{\partial^2 u}{\partial x^2}. \tag{6.18}$$

We briefly comment on the potential benefits of solving the PDE numerically in the transformed versus original coordinates.

Some discretisation schemes are taylored to constant coefficient PDEs or even more specifically to the heat equation. They may achieve better accuracy by exploiting this special form, see, e.g., the comments at the end of 4.2.2 or Exercise 3 in 4.4.

A computational disadvantage of going to log-price is that the transformation maps $S = 0$ *to* $X = -\infty$*, such that the numerical range needs to be truncated at some* X_{min}*, requiring an (additional) asymptotic boundary condition. Another side effect is that a uniform grid in* X *coordinates, say* $-N\Delta x, \ldots, -\Delta x, 0, \Delta x, \ldots, N\Delta x$*,* $x_n = -n\Delta X$*, maps to a very distorted grid in* S *coordinates via* $S_n = \exp(x_n)$*, with sparse points for large* S *and dense points for small* S*. This is likely to be advantageous for large* S*, and inefficient for small* S*, as in both regions the solution is usually nearly linear and can be resolved accurately on a coarse grid.*

The bottom line is that the majority of PDEs we want to solve numerially does not admit a transformation to the heat equation, and for those that do, a "semi-analytic" solution (i.e., closed-form modulo evaluation of an integral) is available and no numerical scheme needed. We therefore avoid making such simplifying transformations but solve the equation in "native" coordinates.

6.1.2 Discretisation

We focus in this section on the example of pricing a put option in the Black-Scholes model, as introduced in 6.1.1, i.e. we solve the IBVP (6.14) to (6.16). The key step is again to *discretise* both the solution and the equation, i.e. to approximate the solution by a finite set of unknowns, and to approximate the continuous equation by a finite-dimensional (linear) system of equations for these values.

Define a grid of $N+1$ nodes $S_0 = 0, S_1 = \Delta S, S_2 = 2\Delta S, \ldots, S_N = N\Delta S = S_{max}$, $\Delta S = S_{max}/N$, further M time steps $t_0 = 0, t_1 = \Delta t, t_2 = 2\Delta t, \ldots, t_M = M\Delta t = T$, $\Delta t = T/M$, at which we introduce a numerical approximation V_n^m to $V(n\Delta S, m\Delta t) = V(S_n, t_m)$. See Fig. 6.1.

The terminal condition (6.16) becomes

$$V_n^M = \max(K - S_n, 0) = \max(K - n\Delta S, 0). \tag{6.19}$$

The upper boundary condition is given by (6.15), so

$$V_N^m = 0.$$

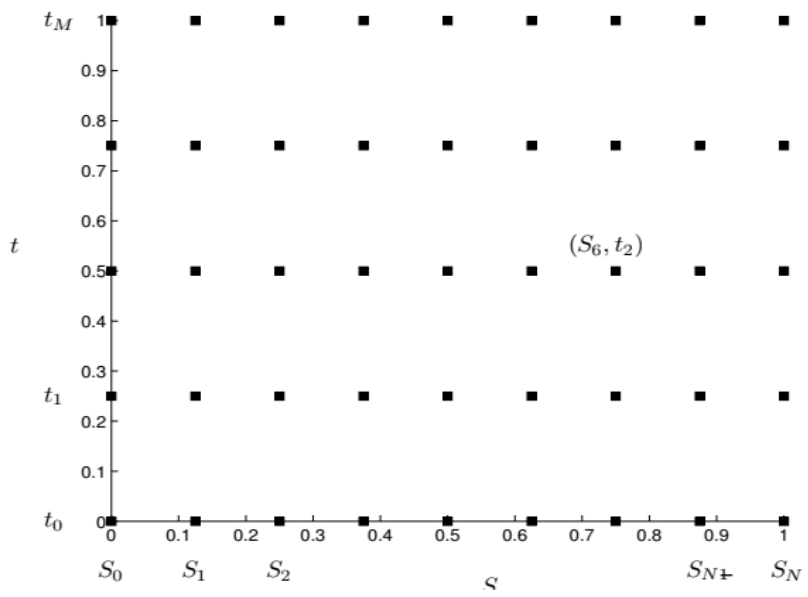

Figure 6.1: Grid, $N = 8$, $M = 4$.

Next, approximate derivatives in the PDE (6.14) by finite differences at grid points, assuming V is sufficiently smooth. We choose for the first derivative a *central difference*

$$\frac{\partial V}{\partial S}(S_n, t_m) = \frac{V(S_{n+1}, t_m) - V(S_{n-1}, t_m)}{2\Delta S} + O(\Delta S^2),$$

and for the second derivative the *second central difference*

$$\frac{\partial^2 V}{\partial S^2}(S_n, t_m) = \frac{V(S_{n+1}, t_m) - 2V(S_n, t_m) + V(S_{n-1}, t_m)}{\Delta S^2} + O(\Delta S^2).$$

For the time derivative, we will consider the *backward difference*

$$\frac{\partial V}{\partial t}(S_n, t_m) = \frac{V(S_n, t_m) - V(S_n, t_{m-1})}{\Delta t} + O(\Delta t)$$

and the *forward difference*

$$\frac{\partial V}{\partial t}(S_n, t_m) = \frac{V(S_n, t_{m+1}) - V(S_n, t_m)}{\Delta t} + O(\Delta t),$$

and combinations thereof. We study specific examples first.

Example 6.1.2 (Explicit Euler for Black-Scholes)**.** *The backward difference leads to the scheme*

$$\frac{V_n^m - V_n^{m-1}}{\Delta t} + \frac{1}{2}\sigma^2 \Delta S^2 n^2 \frac{V_{n+1}^m - 2V_n^m + V_{n-1}^m}{\Delta S^2} + r\Delta S n \frac{V_{n+1}^m - V_{n-1}^m}{2\Delta S} - rV_n^m = 0,$$

$m = 1, \ldots, M$, $n = 1, \ldots, N-1$. *Here* V_n^{m-1} *can be computed explicitly from the values* V_{n-1}^m, V_n^m, V_{n+1}^m *as*

$$V_n^{m-1} = A_n^m V_{n-1}^m + B_n^m V_n^m + C_n^m V_{n+1}^m, \tag{6.20}$$

where

$$A_n^m = \frac{1}{2}n^2\sigma^2\Delta t - \frac{1}{2}nr\Delta t, \tag{6.21}$$

$$B_n^m = 1 - n^2\sigma^2\Delta t - r\Delta t, \tag{6.22}$$

$$C_n^m = \frac{1}{2}n^2\sigma^2\Delta t + \frac{1}{2}nr\Delta t. \tag{6.23}$$

This is the explicit Euler *scheme. The backward difference is explicit for the backward equation, as the forward difference is explicit for the forward equation.*

At $n = 0$*, the scheme naturally reduces to*

$$V_0^{m-1} = (1 - r\Delta t)V_0^m, \tag{6.24}$$

and at the upper boundary we know

$$V_N^{m-1} = 0.$$

This procedure can be applied inductively in m*, starting from* V_n^M *given by (6.19), to compute all values down to* $m = 0$*, i.e.* $t = 0$*. In the Black-Scholes case, with parameters constant over time,* A_n^m*,* B_n^m *and* C_n^m *do not actually depend on* m*, and therefore do not need to be recomputed in each timestep.*

Example 6.1.3 (Implicit Euler for Black-Scholes)**.** *The forward difference leads to the scheme*

$$\frac{V_n^{m+1} - V_n^m}{\Delta t} + \frac{1}{2}\sigma^2 \Delta S^2 n^2 \frac{V_{n+1}^m - 2V_n^m + V_{n-1}^m}{\Delta S^2} + r\Delta S n \frac{V_{n+1}^m - V_{n-1}^m}{2\Delta S} - rV_n^m = 0,$$

$m = 0, \ldots, M-1$, $n = 1, \ldots, N-1$. *Here* V_n^m *is implicitly given by the system of equations*

$$a_n^m V_{n-1}^m + b_n^m V_n^m + c_n^m V_{n+1}^m = V_n^{m+1}, \tag{6.25}$$

where

$$\begin{aligned} a_n^m &= -\frac{1}{2}n^2\sigma^2\Delta t + \frac{1}{2}nr\Delta t, \\ b_n^m &= 1 + n^2\sigma^2\Delta t + r\Delta t, \\ c_n^m &= -\frac{1}{2}n^2\sigma^2\Delta t - \frac{1}{2}nr\Delta t. \end{aligned}$$

This is the implicit Euler *scheme. The forward difference is implicit for the backward equation, as the backward difference is implicit for the forward equation. At $n = 0$, the scheme naturally reduces to*

$$(1 + r\Delta t)V_0^m = V_0^{m+1},$$

and at the upper boundary, we know

$$V_N^m = 0.$$

In each time step, $m = M - 1, \ldots, 0$, a linear system for the unknowns V_0^m, V_1^m, ..., V_{N-1}^m has to be solved.

Following these examples, we deduce the equivalent of the θ-timestepping scheme from earlier for the backward PDE

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2(S,t)\frac{\partial^2 V}{\partial S^2} + \mu(S,t)\frac{\partial V}{\partial S} - r(S,t)V = 0.$$

Recall the θ-timestepping scheme as interpreted as an explicit step of size $(1-\theta)\Delta t$, followed by a fully implicit step if size $\theta\Delta t$, with $\theta \in [0,1]$. One thus gets

$$\delta_t^- V_n^m + \left(\frac{1}{2}\sigma^2(S_n, t_{m-\theta})\delta_S^2 + \mu(S_n, t_{m-\theta})\delta_S - r(S_n, t_{m-\theta})\right)(\theta V_n^{m-1} + (1-\theta)V_n^m) = 0, \quad (6.26)$$

where δ_S and δ_S^2 are first and second central difference operators, e.g. $\delta_S V_n^m = (V_{n+1}^m - V_{n-1}^m)/2\Delta S$ etc, and the backward time difference $\delta_t^- V_n^m = (V_n^m - V_n^{m-1})/\Delta t$. This defines a scheme backwards in time, which can be written as

$$a_n^m V_{n-1}^{m-1} + b_n^m V_n^{m-1} + c_n^m V_{n+1}^{m-1} = A_n^m V_{n-1}^m + B_n^m V_n^m + C_n^m V_{n+1}^m, \quad (6.27)$$

where

$$a_n^m = -\frac{1}{2}\theta\Delta t\left(\sigma^2(S_n, t_{m-\theta})/\Delta S^2 - \mu(S_n, t_{m-\theta})/\Delta S\right) \quad (6.28)$$

$$b_n^m = 1 + \theta\Delta t\left(\sigma^2(S_n, t_{m-\theta})/\Delta S^2 + r(S_n, t_{m-\theta})\right) \quad (6.29)$$

$$c_n^m = -\frac{1}{2}\theta\Delta t\left(\sigma^2(S_n, t_{m-\theta})/\Delta S^2 + \mu(S_n, t_{m-\theta})/\Delta S\right) \quad (6.30)$$

and

$$A_n^m = \frac{1}{2}(1-\theta)\Delta t\left(\sigma^2(S_n,t_{m-\theta})/\Delta S^2 - \mu(S_n,t_{m-\theta})/\Delta S\right) \quad (6.31)$$

$$B_n^m = 1-(1-\theta)\Delta t\left(\sigma^2(S_n,t_{m-\theta})/\Delta S^2 + r(S_n,t_{m-\theta})\right) \quad (6.32)$$

$$C_n^m = \frac{1}{2}(1-\theta)\Delta t\left(\sigma^2(S_n,t_{m-\theta})/\Delta S^2 + \mu(S_n,t_{m-\theta})/\Delta S\right) \quad (6.33)$$

for $n = 1,\ldots,N-1$. Setting $\theta = 0$, one recovers the explicit method, with $\theta = 1$ the fully implicit method. The choice $\theta = 0.5$ is the second order accurate Crank-Nicolson method.

Boundary conditions depend on the problem at hand. For $n = 0$, in the Black-Scholes case, $a_n^m = c_n^m = A_n^m = C_n^m = 0$. The discretisation of the PDE naturally reduces to a discretisation of the ODE (6.9) and no numerical boundary conditions are necessary. For $n = N$, for the put, set $V_N^m = V_N^{m-1} = 0$ and eliminate them from the other equations, then the linear system takes the form

$$\underbrace{\begin{pmatrix} b_0^m & c_0^m & & \ldots & 0\\ a_1^m & b_1^m & c_1^m & \ldots & 0\\ 0 & \ddots & \ddots & \ddots & 0\\ 0 & \ldots & a_{N-2}^m & b_{N-2}^m & c_{N-2}^m\\ 0 & \ldots & 0 & a_{N-1}^m & b_{N-1}^m \end{pmatrix}}_{:=K_1(\theta)} \begin{pmatrix} V_0^{m-1}\\ V_1^{m-1}\\ \vdots\\ V_{N-2}^{m-1}\\ V_{N-1}^{m-1}\end{pmatrix} = \underbrace{\begin{pmatrix} B_0^m & C_0^m & & \ldots & 0\\ A_1^m & B_1^m & C_1^m & \ldots & 0\\ 0 & \ddots & \ddots & \ddots & 0\\ 0 & \ldots & A_{N-2}^m & B_{N-2}^m & C_{N-2}^m\\ 0 & \ldots & 0 & A_{N-1}^m & B_{N-1}^m \end{pmatrix}}_{:=K_0(\theta)} \begin{pmatrix} V_0^m\\ V_1^m\\ \vdots\\ V_{N-2}^m\\ V_{N-1}^m\end{pmatrix}. \quad (6.34)$$

This leads to Algorithm 3. The form of (6.34) is special because of degeneracy of the PDE

Algorithm 3 θ-method in European option pricing

1: Choose S_{max}, N, M, θ
2: $\Delta S = S_{max}/N$, $\Delta t = T/M$
3: $S \leftarrow (0,\Delta S,\ldots,(N-1)\Delta S) \in \mathbb{R}^N$
4: $V \leftarrow \text{payoff}(S) \in \mathbb{R}^N$
5: Set up matrices K_0 and $K_1 \in \mathbb{R}^{N\times N}$ as per (6.34) and (6.28)–(6.33)
6: **for** $m \leftarrow M$, 1 **do**
7: $\quad rhs \leftarrow K_0 V$
8: $\quad$ Solve $K_1 V = rhs$ for V
9: **end for**

coefficients at the lower boundary, and zero boundary conditions at the upper boundary.

Anticipating an example from section 6.4 on other payoffs, for a call option, one can envoce put-call parity (6.79) to deduce

$$V(S_{max},t) \approx S_{max} - K\mathrm{e}^{-r(T-t)},$$

where the approximation error is exactly identical to the put. The numerical boundary condition at S_N is then $V_N^m = S_N - K\mathrm{e}^{-r(T-t_m)}$. Proceding similar to the put leads to a system

$$K_1(\theta)V^{m-1} = K_0(\theta)V^m + F^m,$$

where the "boundary term" F_n^m is zero for $n = 0,1,\ldots,N-2$ and

$$F_{N-1}^m = -c_{N-1}V_N^{m-1} + C_{N-1}^m V_N^m.$$

The reason why there is no contribution from the $S = 0$ boundary is that $\mu(0, t) = \sigma(0, t) = 0$. A more detailed discussion follows in 6.4.

6.1.3 Numerical tests

We investigate the explicit scheme first, on the example of a Black-Scholes put. It is clearly a special case of Algorithm 3, but the main motivation for using the explict scheme is usually its ease of implementation, and for that reason we formulate the scheme very explicitly (pardon the pun) in Algorithm 4. Note that as the coefficients A_n^m, B_n^m, C_n^m do not depend on m

Algorithm 4 Explicit Euler scheme

$\sigma \leftarrow 0.4$, $r \leftarrow 0.05$
$T \leftarrow 1$, $K \leftarrow 0.25$
$S_{max} \leftarrow 1$
$N \leftarrow 16$, $M \leftarrow 16$
$\Delta t \leftarrow T/M$, $\Delta S \leftarrow S_{max}/N$
for $n \leftarrow 0$, N **do**
    $A_n \leftarrow \frac{1}{2}n^2\sigma^2\Delta t - \frac{1}{2}nr\Delta t$
    $B_n \leftarrow 1 - n^2\sigma^2\Delta t - r\Delta t$
    $C_n \leftarrow \frac{1}{2}n^2\sigma^2\Delta t + \frac{1}{2}nr\Delta t$
end for
for $n \leftarrow 0$, N **do**
    $V_n^M \leftarrow \max(K - n\Delta S, 0)$
end for
for $m \leftarrow M$, 1 **do**
    $V_0^{m-1} \leftarrow B_0V_0^m + C_0V_1^m$
    for $n \leftarrow 1$, $N - 2$ **do**
        $V_n^{m-1} \leftarrow A_nV_{n-1}^m + B_nV_n^m + C_nV_{n+1}^m$
    end for
    $V_{N-1}^{m-1} \leftarrow A_{N-1}V_{N-2}^m + B_{N-1}V_{N-1}^m$
end for

for this example, because the coefficients in the Black-Scholes equation are time-independent, they need only be computed once. The boundary condition $V_N^m = 0$ is applied by leaving out the term $C_{N-1}V_N^m$ in the equation for V_{N-1}^{m-1}.

In Fig. 6.2, the numerical solution V_n^0 is compared to the analytically known Black-Scholes price $V(S, 0)$ for $N = 16$ and $M = 16$.

To reduce the error, try and increase the number of time steps, say $M = 32$, still with $N = 16$. As shown in Fig. 6.3, the error does not decrease, as a matter of fact it becomes slightly larger. We therefore suspect that the error is mainly due to the space discretisation and increase the number of space steps as well, e. g. $N = 32$, $M = 32$. However, this goes terribly wrong and the numerical solution blows up.

To investigate this systematically, we study the error over grid refinements in Table 6.1.

Table 6.1 shows that reasonable solutions are obtained only provided that the number of time steps is sufficiently large, in particular increasing with the number of grid points. The

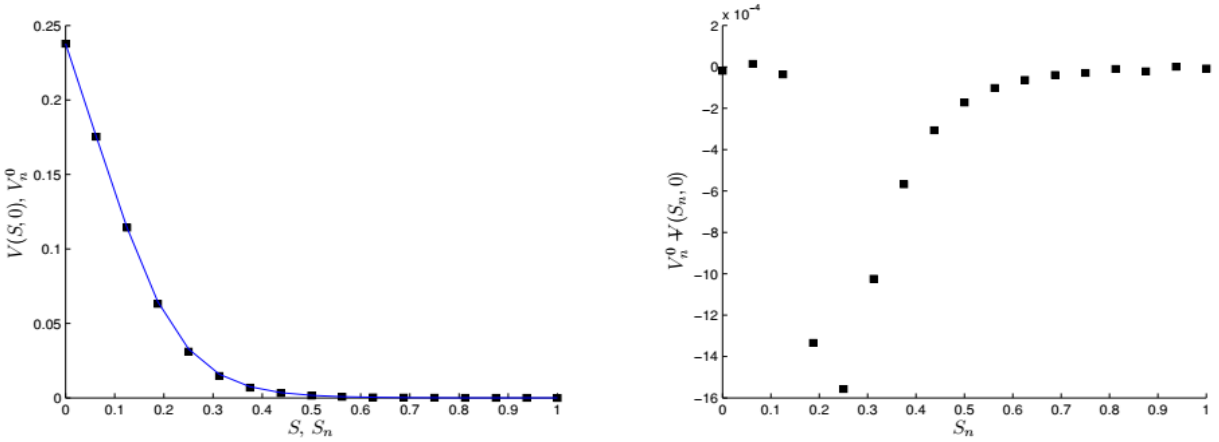

Figure 6.2: Left: Black-Scholes value of a put and finite difference approximation with 16 time steps and 16 grid intervals. Right: Error at the grid points. Parameters used are $\sigma = 0.4$, $r = 0.05$, $T = 1$, $K = 0.25$, $S_{max} = 1$.

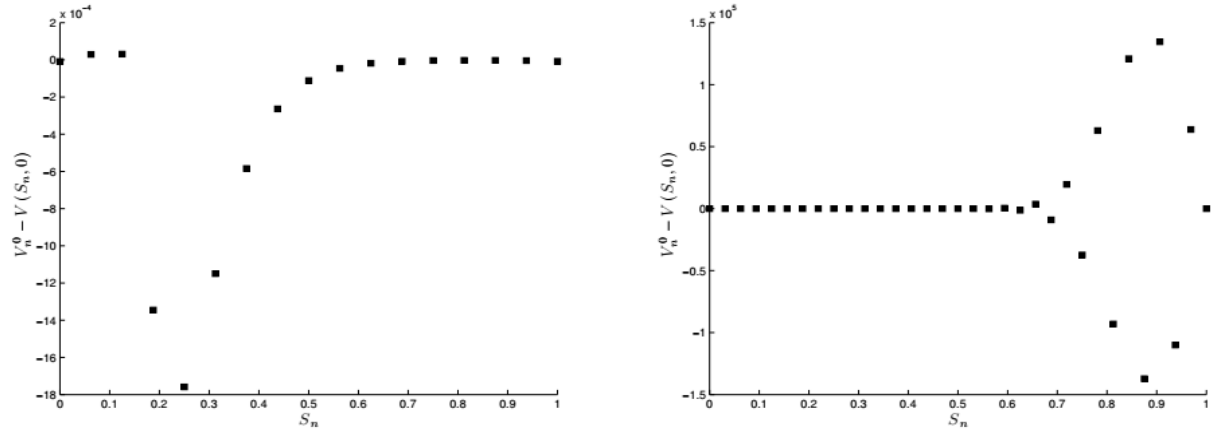

Figure 6.3: Finite difference error for $N = 16$, $M = 32$ (left) and $M = 32$, $N = 32$ (right).

behaviour therefore depends on the path of space/time refinement, which us referred to as *conditional stability*. Specifically, it appears two time refinements are required per space refinement.

If we look at the italic values in the "diagonal", obtained by two time refinements per refinement in the space coordinate, the error goes down by a factor of roughly four. The tabular indicates that we get second order convergence in space, first order convergence in time,

$$\epsilon = O(\Delta t) + O(\Delta S^2). \tag{6.35}$$

Along this "diagonal", $N^2/M = \Delta t/\Delta S^2 = 4$. It is clear that stability can only depend on the dimensionless variable $\sigma^2 \Delta t$. The explicit scheme appears stable as long as

$$\sigma^2 \frac{\Delta t}{\Delta S^2} < 0.64. \tag{6.36}$$

We now contrast this with the Crank-Nicolson scheme in Table 6.2.

$M\backslash N$	16	32	64	128	256	512
16	-1.5569e-03	-7.2855e-04	-1.8888e+07	-1.5937e+17	-5.0711e+26	-1.2048e+36
32	-1.7585e-03	-2.8995e-04	-3.3148e+09	-2.4715e+28	-3.2423e+47	-3.4126e+66
64	*-1.8596e-03*	-3.7393e-04	-7.2127e+13	-6.6809e+44	-4.0451e+80	-1.5998e+118
128	-1.9102e-03	-4.1590e-04	-7.3410e+30	-5.9686e+75	-1.0071e+135	-5.0670e+204
256	-1.9355e-03	*-4.3688e-04*	-6.9726e+54	-2.2570e+151	-3.7686e+238	-Inf
512	-1.9482e-03	-4.4736e-04	-1.0281e-04	NaN	NaN	NaN
1024	-1.9545e-03	-4.5261e-04	*-1.0789e-04*	NaN	NaN	NaN
2048	-1.9576e-03	-4.5523e-04	-1.1044e-04	6.8825e+140	NaN	NaN
4096	-1.9592e-03	-4.5654e-04	-1.1171e-04	*-2.6895e-05*	NaN	NaN
8192	-1.9600e-03	-4.5719e-04	-1.1235e-04	-2.7526e-05	NaN	NaN
16384	-1.9604e-03	-4.5752e-04	-1.1266e-04	-2.7842e-05	*-6.7188e-06*	NaN
32768	-1.9606e-03	-4.5768e-04	-1.1282e-04	-2.8000e-05	-6.8764e-06	NaN
65536	-1.9607e-03	-4.5777e-04	-1.1290e-04	-2.8079e-05	-6.9552e-06	*-1.6794e-06*

Table 6.1: Finite difference error for the explicit scheme, at-the-money, i. e. at $S = K$, for M time steps and N grid intervals. The values given in the table are $V_k^0 - V(K, 0)$, where k is the index for which $S_k = K$.

$M\backslash N$	16	32	64	128	256	512
16	-1.9534e-03	-4.5252e-04	*-1.0792e-04*	-5.0050e-05	-2.8698e-04	-5.0914e-04
32	-1.9590e-03	-4.5651e-04	-1.1171e-04	*-2.6906e-05*	-1.9418e-05	-1.4315e-04
64	-1.9603e-03	-4.5751e-04	-1.1266e-04	-2.7844e-05	*-6.7223e-06*	-8.2854e-06
128	-1.9607e-03	-4.5776e-04	-1.1290e-04	-2.8079e-05	-6.9559e-06	*-1.6804e-06*
256	-1.9608e-03	-4.5783e-04	-1.1296e-04	-2.8138e-05	-7.0144e-06	-1.7387e-06
512	-1.9608e-03	-4.5784e-04	-1.1298e-04	-2.8153e-05	-7.0291e-06	-1.7533e-06

Table 6.2: Finite difference error for the Crank-Nicolson scheme, at-the-money, i. e. at $S = K$, for M time steps and N grid intervals. The values given in the table are $V_k^0 - V(K, 0)$, where k is the index for which $S_k = K$.

If we look at the tabular line-wise, the error initially decays roughly by a facor of four from one column to the next, before it levels off and in fact increases slightly, but there is no sense of explosion. We see a similar effect looking down the columns. The Crank-Nicolson scheme appears of second order accurate in both ΔS and Δt,

$$\epsilon = O(\Delta t^2) + O(\Delta S^2), \tag{6.37}$$

and unconditionally stable. The smallest errors over the columns are highlighted in italic. This indicates the optimal refinement strategy as one where $N = 4M$.

6.1.4 Complexity considerations

The ultimate measure in judging the efficiency of a numerical scheme for a particular problem are the computational resources needed to solve the problem to a certain desired accuracy. In the wider context, issues like the implementation time, re-usability, availability of components etc will be relevant, but we here focus on computational time, specifically for the solution of the Black-Scholes PDE. Memory usage would be another factor but is typically less critical in this context.

Denote by the ϵ-*complexity* $C(\epsilon)$ the computational cost required to solve the problem to accuracy ϵ. We measure cost here as computation time. Clearly, the precise time will depend on factors like the implementation, computing architecture, and via the error on the parameters of the problem. As a useful crude measure it is sufficient to look at the *order* of $C(\epsilon)$ for small ϵ. This measure does not necessarily tell us which of two algorithms is superior for a given accuracy level, but assuming we want an accurate solution typically does give a good indication of the relative cost.

There are two ingredients to the ϵ-complexity. Firstly, the numerical parameters, here number of grid points N and time steps M, needed to solve the problem to accuracy ϵ. This has to come out of the numerical analysis of the method and assumes that we are able to determine N and M in such a way that the error is below ϵ. And secondly, the complexity of solving the problem for numerical parameters N and M.

The second question is usually easier to answer. By inspection of Algorithm 4, for the explicit Euler finite difference scheme, the number of simple operations is proportional to M via the time step loop and N via the loop over grid points within each timestep, $C \sim NM$.

From (6.35), we see that we want to choose $\Delta t \sim \Delta S^2$ for the explicit Euler scheme to balance error terms resulting from the time and asset discretisation. In fact, from (6.36), this is also necessary for stability. An optimal choice is therefore

$$M \sim N^2 \sim \epsilon^{-1},$$

so, inserting back in, for explicit Euler,

$$C_{EE}(\epsilon) = O(\epsilon^{-3/2}).$$

For Crank-Nicolson, inspecting Algorithm 3, the computational time is still proportional to NM if the linear system can be solved in $O(N)$ operations in each timesteps. The Thomas algorithm does this. The error is now given by (6.37) and we want to choose

$$M \sim N \sim \epsilon^{-1/2},$$

so, inserting again,

$$C_{CN}(\epsilon) = O(\epsilon^{-1}).$$

The Crank-Nicolson method is more efficient asymptotically because for roughly the same computational cost for identical grid parameters, higher accuracy is achieved. This means conversely that less timesteps are needed for the same accuracy which makes the method computationally cheaper.