Further Finite Difference Methods

In this lecture. . .

- Review of explicit scheme and introduction to stability analysis
- implicit finite-difference methods including Crank–Nicolson
- Review of numerical linear algebra
- The $\theta-$method
- American-style exercise
- Numerical schemes for Exotic options

- About the explicit finite-difference method for two-factor models

- The ADI methods

1 Introduction

There are many more ways of solving parabolic partial differential equations than the explicit method.

The more advanced methods are usually more complicated to program but have advantages in terms of stability and speed.

This session presents a class of numerical methods for derivative pricing models that are given in the form of linear parabolic partial differential equations.

We begin with a brief review of the explicit scheme in a more mathematical sense.

2 Model Problem

Consider the following Black-Scholes pricing problem for the value of a European Call Option $V = V(S,t)$:

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

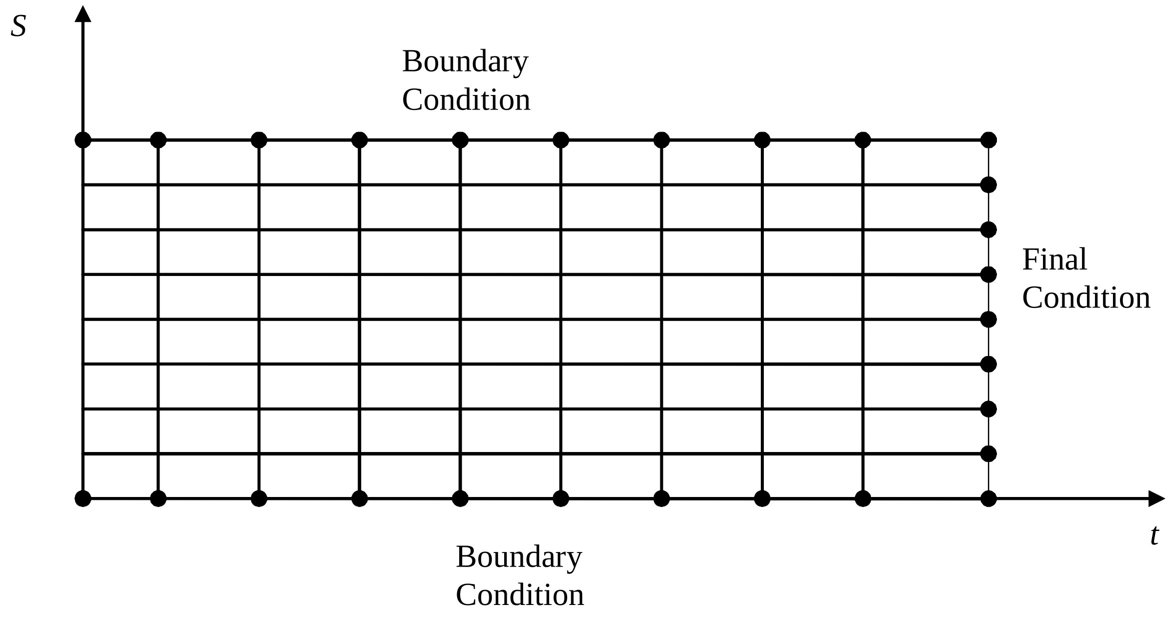
$$\begin{aligned} V(0,t) &= 0 \\ \lim_{S\to\infty} V(S,t) &\to S \\ V(S,T) &= \max(S-E,0)\,, \\ S &\in [0,\infty)\,, \quad t \in (0,T) \end{aligned}$$

where S is the spot price of the underlying financial asset, t is the time, $E > 0$ is the strike price, T the expiry date, $r \geq 0$ the interest rate, D is the dividend yield and σ is the volatility of S.

This is an Initial Value Problem (IVP). It consists of a parabolic pde (similar to a diffusion equation) which is first order in time t and second order in the asset price S. It is solved using two boundary conditions at $S = 0$ and the limit $S \to \infty$. The initial condition which is in fact a *final condition*, in this case, is defined at the end of the time horizon $t = T$. So knowing the value of V for the final time T allows us to time-step backwards to the current (starting) time $t = 0$ and hence calculate the option value for all values of t.

There are two steps to solving a PDE numerically using the FDM. Firstly the partial derivative terms in the equation are replaced by approximations. Secondly the region over which the equation is to be solved is decomposed into an area consisting of vertical and horizontal lines to produce a grid. The equation is then solved at each of the grid points.

The problem is to be solved over the following region:



This is to be solved by a Finite Difference (FD) Scheme.

By expressing $S = n\delta S$ and $t = m\delta t$, we will obtain a difference equation for the Black-Scholes equation.

$V(S,t) = V(n\delta S,\ m\delta t) = V_n^m$. $\delta S = 1/S^*$ where $S^* \gg E$ is a suitably large value of S ; $\delta t = 1/T$.

Take N and M steps for S and t respectively, so

$$\begin{aligned} S &= n\delta S \qquad 0 \le n \le N \\ t &= m\delta t \qquad 0 \le m \le M. \end{aligned}$$

2.1 Taylor Series Approximations

We begin with the Taylor-series approximations for the 1st and 2nd order derivatives.

If $V = V(S,t)$ and S & t both change by an amount δS and δt in turn, so that $S \to S + \delta S$ and $t \to t + \delta t$ then the change in V can be obtained using a two-dimensional Taylor expansion

$$V(S + \delta S,\ t + \delta t) =$$

$$V(S,t) + \frac{\partial V}{\partial S}\delta S + \frac{\partial V}{\partial t}\delta t + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\delta S^2$$
$$+\frac{1}{2}\frac{\partial^2 V}{\partial t^2}\delta t^2 + \frac{\partial^2 V}{\partial S \partial t}\delta S \delta t + O\left(\delta S^3,\ \delta t^3\right)$$

Consider

$$V(S,\ t + \delta t) = V(S,t) + \frac{\partial V}{\partial t}\delta t + O\left(\delta t^2\right)$$

and rearranging gives a forward difference

$$\frac{\partial V}{\partial t} = \frac{V(S, t + \delta t) - V(S,t)}{\delta t} + O(\delta t)\,.$$

Let us use a backward time difference

$$\frac{\partial V}{\partial t} = \frac{V(S,t) - V(S,t-\delta t)}{\delta t} + O(\delta t),$$

which becomes, in finite difference form

$$\frac{\partial V}{\partial t}(n\delta S,\ m\delta t) \sim \frac{V_n^m - V_n^{m-1}}{\delta t}.$$

We now derive approximations for derivative terms involving S. Start by considering V at $S+\delta S$ and $S-\delta S$

$$V(S+\delta S,\ t) = V(S,t) + \frac{\partial V}{\partial S}\delta S + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\delta S^2 + O\left(\delta S^3\right) \tag{1}$$

$$V(S-\delta S,\ t) = V(S,t) - \frac{\partial V}{\partial S}\delta S + \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\delta S^2 - O\left(\delta S^3\right) \tag{2}$$

(1) − (2) gives

$$V(S+\delta S,\ t) - V(S-\delta S,\ t) = 2\frac{\partial V}{\partial S}\delta S + O\left(\delta S^3\right)$$

which upon rearranging and using finite difference notation yields

$$\frac{\partial V}{\partial S}(n\delta S,\ m\delta t) = \frac{V_{n+1}^m - V_{n-1}^m}{2\delta S} + O\left(\delta S^2\right)$$

and hence giving us a scheme for the first derivative $\frac{\partial V}{\partial S}$.

$(1) + (2)$ gives

$$V(S+\delta S, t) + V(S-\delta S, t) = 2V(S,t) + \frac{\partial^2 V}{\partial S^2}\delta S^2 + O\left(\delta S^4\right)$$

and hence

$$\frac{\partial^2 V}{\partial S^2} = \frac{V(S+\delta S,\ t) - 2V(S,t) + V(S-\delta S,\ t)}{\delta S^2} + O\left(\delta S^2\right)$$

and hence a finite difference approximation for the second derivative

$$\frac{\partial^2 V}{\partial S^2}(n\delta S,\ m\delta t) = \frac{V_{n-1}^m - 2V_n^m + V_{n+1}^m}{\delta S^2} + O\left(\delta S^2\right)$$

$$\begin{aligned} \frac{\partial V}{\partial t} &\sim \frac{V_n^m - V_n^{m-1}}{\delta t}, \qquad \frac{\partial V}{\partial S} \sim \frac{V_{n+1}^m - V_{n-1}^m}{2\delta S}, \\ \frac{\partial^2 V}{\partial S^2} &\sim \frac{V_{n-1}^m - 2V_n^m + V_{n+1}^m}{\delta S^2} \end{aligned}$$

By expressing $S = n\delta S$ and $t = m\delta t$, we will obtain a difference equation for the Black-Scholes equation.

$V(S,t) = V(n\delta S,\ m\delta t) = V_n^m$. $\delta S = 1/S^*$ where $S^* \gg E$ is a suitably large value of S ; $\delta t = 1/T$.

Take N and M steps for S and t respectively, so

$$\begin{aligned} S &= n\delta S \qquad 0 \le n \le N \\ t &= m\delta t \qquad 0 \le m \le M. \end{aligned}$$

and

$$\begin{aligned} \frac{\partial V}{\partial t} &\sim \frac{V_n^m - V_n^{m-1}}{\delta t}, \\ \frac{\partial V}{\partial S} &\sim \frac{V_{n+1}^m - V_{n-1}^m}{2\delta S}, \\ \frac{\partial^2 V}{\partial S^2} &\sim \frac{V_{n-1}^m - 2V_n^m + V_{n+1}^m}{\delta S^2}. \end{aligned} \qquad (3)$$

Substituting (3) in the BSE gives

$$\begin{aligned} & \tfrac{V_n^m - V_n^{m-1}}{\delta t} + \tfrac{1}{2}n^2\sigma^2\left(V_{n-1}^m - 2V_n^m + V_{n+1}^m\right) + \\ & \tfrac{1}{2}(r-D)\,n\left(V_{n+1}^m - V_{n-1}^m\right) - rV_n^m \\ = \ & 0 \end{aligned}$$

and rearrange to obtain a *backward marching* scheme in time

$$\begin{aligned} V_n^{m-1} &= V_n^m + \delta t \left(\frac{1}{2} n^2 \sigma^2 \left(V_{n-1}^m - 2V_n^m + V_{n+1}^m \right) \right) \\ &\quad + \delta t \left(\frac{1}{2} (r-D)\, n \left(V_{n+1}^m - V_{n-1}^m \right) - r V_n^m \right) \\ &\equiv F\left(V_{n-1}^m,\ V_n^m,\ V_{n+1}^m \right) \end{aligned}$$

Now for the RHS collect coefficients of each variable term V, to get

$$V_n^{m-1} = \alpha_n V_{n-1}^m + \beta_n V_n^m + \gamma_n V_{n+1}^m \qquad (4)$$

where

$$\begin{aligned} \alpha_n &= \frac{1}{2} \left(n^2 \sigma^2 - n(r-D) \right) \delta t, \qquad (5) \\ \beta_n &= 1 - \left(r + n^2 \sigma^2 \right) \delta t, \\ \gamma_n &= \frac{1}{2} \left(n^2 \sigma^2 + n(r-D) \right) \delta t \end{aligned}$$

(4) is a linear difference equation. We will use this to march backwards in time, i.e. given a solution at time step m we can use (4) to approximate a solution at the

next time step $(m-1)$.The difference equation (4) is not valid at the boundaries.

Boundary conditions:

At $S = 0$, i.e. $n = 0$, the BSE becomes

$$\begin{aligned} \frac{\partial V}{\partial t} &= rV \Rightarrow \\ V_0^{m-1} &= (1 - r\delta t) V_0^m \end{aligned}$$

This also follows from

$$\alpha_0 = 0 = \gamma_0, \quad \beta_0 = 1 - r\delta t.$$

As S becomes very large, i.e. $S \to \infty$ (S^*) the probability of it becoming lower than the Exercise becomes negligible, therefore $\Delta = \Delta(t)$ only, hence $\Gamma \to 0$.

The problem arises at $n = N$. We cannot use our difference equation at the boundary, as we end up with a term V_{N+1}^m, which is not defined. So we use the gamma condition mentioned above.

We know $\Gamma \sim \dfrac{V_{n-1}^m - 2V_n^m + V_{n+1}^m}{\delta S^2} = 0$

which upon rearranging gives at $n = N$

$$V_{N+1}^m = 2V_N^m - V_{N-1}^m$$

and substituting in the difference equation gives

$$V_N^{m-1} = (\alpha_N - \gamma_N)\, V_{N-1}^m + (\beta_N + 2\gamma_N)\, V_N^m.$$

In summary, the scheme is

$$\left.\begin{array}{c} V_n^{m-1} = \alpha_n V_{n-1}^m + \beta_n V_n^m + \gamma_n V_{n+1}^m \\ M > m \geq 1; \quad 1 \leq n \leq N-1 \end{array}\right\} \text{ D.E}$$

$$\left.\begin{array}{c} V_n^M = \max(n\delta S - E, 0) \\ 0 \leq n \leq N; \end{array}\right\} \text{ Final Payoff Condition}$$

$$\left.\begin{array}{c} V_0^{m-1} = \beta_0 V_0^m \\ M \geq m \geq 1 \end{array}\right\} \text{BC at } \ (S = 0)$$

$$\left.\begin{array}{c} V_N^{m-1} = (\alpha_N - \gamma_N)\, V_{N-1}^m + (\beta_N + 2\gamma_N)\, V_N^m \\ M \geq m \geq 1; \quad S = N\delta S \end{array}\right\} \text{BC at } \ S^*$$

3 Fourier Stability (Von Neumann's) Method

A method is called step-wise unstable if for a fixed grid (i.e. $\delta t,\ \delta S$ constant) there exists an initial perturbation which "blows up" as $t \to \infty$, i.e. as we march in time. Here in a backward marching scheme we have $t \to 0$ $(m \to 0)$. The question we wish to answer is "do small errors propagate along the grid and grow exponentially?". We hope not!

Assume an initial disturbance which is proportional to $\exp(in\omega)$. We therefore study the propagation of perturbations created at any given point in time.

If $\widehat{V}_n^m$ is an approximation to the exact solution ${V_n}^m$ then

$$\widehat{V}_n^m = {V_n}^m + {E_n}^m$$

where ${E_n}^m$ is the associated error. Then ${E_n}^m$ also satisfies the difference equation (4) to give

$$E_n^{m-1} = \alpha_n E_{n-1}^m + \beta_n E_n^m + \gamma_n E_{n+1}^m.$$

Put

$$E_n^m = \overline{a}^m \exp(in\omega) \qquad (6)$$

which is oscillatory of amplitude $\overline{a}$ and frequency ω. Substituting (6) into (4) gives

$$\overline{a}^{m-1} e^{(in\omega)} = \alpha_n \overline{a}^m e^{i(n-1)\omega} + \beta_n \overline{a}^m e^{in\omega} + \gamma_n \overline{a}^m e^{i(n+1)\omega}$$

which becomes

$$\overline{a}^{-1} = \alpha_n e^{-i\omega} + \beta_n + \gamma_n e^{i\omega}.$$

Now stability criteria arises from the balancing of the time dependency and diffusion terms, so that

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} = 0$$

From (5) we take the following contributions

$$\alpha_n = \frac{1}{2} n^2 \sigma^2 \delta t, \quad \beta_n = 1 - n^2 \sigma^2 \delta t, \quad \gamma_n = \frac{1}{2} n^2 \sigma^2 \delta t$$

$$\begin{aligned} \overline{a}^{-1} &= \frac{1}{2} n^2 \sigma^2 \delta t \left(e^{i\omega} + e^{-i\omega} \right) + 1 - n^2 \sigma^2 \delta t \\ &= n^2 \sigma^2 \delta t (\cos\omega - 1) + 1. \end{aligned}$$

we have

$$\overline{a}^{-1} = 1 - 2n^2\sigma^2 \sin^2 \frac{\omega}{2}\delta t$$

For stability $\overline{a}^{-1}$ must be bounded, i.e. $\left|\overline{a}^{-1}\right| \leq 1 \Longleftrightarrow$ $|a| \geq 1$ (as it is a backward marching scheme), i.e.

$$\left|1 - 2n^2\sigma^2 \sin^2 \frac{\omega}{2}\delta t\right| \leq 1$$

which upon simplifying we find is

$$\delta t \leq \frac{1}{\sigma^2 N^2} \tag{7}$$

so $\delta t \sim O\left(N^{-2}\right)$.

The beauty of the explicit method lies in its simplicity, both in numerical and computational terms.

However, the main disadvantage is associated with the stability criteria, given by (7).

This condition puts severe constraints on the viability of the method. If it is not satisfied, we will observe exponentially growing oscillations in our numerical solution as we iterate backwards in time.

Given that $\delta t = O\left(\frac{1}{N^2}\right)$, we see that the accuracy can be improved by increasing the number of asset steps N. However doubling N requires the use of four times as many time-steps, to satisfy the stability condition.

4 Early Exercise Feature - American Options

If we can exercise an option during some time interval, before its expiry date, then the *no-arbitrage* argument tells us that the value of the option V can not be less than the payoff $P(S,t)$ during that time period, so

$$V \geq P(S,t).$$

In the explicit scheme, the early exercise constraint can be implemented in a most trivial manner. Consider the time interval T in which the option may be exercised. As we step backwards in time, the option value is computed. If this price is less than the payoff during T, it is set equal to the payoff. So at each time step, we solve the explicit scheme to obtain the option price $\overline{V}$. Then check the condition $\overline{V} < P(S,t)$? If this is true then the option price $V = P(S,t)$, else $V = U$.

This strategy of checking the early exercise constraint is called the *cutoff* method. Thus the explicit FDM can be expressed in compact form as

$$\begin{aligned} U_n^{m-1} &= \alpha_n V_{n-1}^m + \beta_n V_n^m + \gamma_n V_{n+1}^m \\ V_n^{m-1} &= \begin{cases} U_n^{m-1} & \text{if } U_n^{m-1} \geq P_n^{m-1} \\ P_n^{m-1} & \text{if } U_n^{m-1} < P_n^{m-1} \end{cases} \end{aligned}$$

where P_n^m is the FDA for the payoff at $(S,\ t)$, i.e. $P_n^m =$Payoff$(n\delta S,\ m\delta t)$ as opposed to simply P_n^M at time $t = T$. So we have a time dependent payoff function, defined (at each time step) for the life of the option at the time the contract is written.

5 Variable Parameters

There are a number of parameters in the BSE, which need not be constant. FDM can easily handle problems involving non-constant parameters. Suppose the volatility, dividend yield and interest rates are functions of asset price and time, such that the pricing equation becomes

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

The explicit FD scheme now becomes

$$V_n^{m-1} = \alpha_n^m V_{n-1}^m + \beta_n^m V_n^m + \gamma_n^m V_{n+1}^m$$

where

$$\begin{aligned}
\alpha_n &= \frac{1}{2}\left(\left(\sigma_n^m\right)^2 n^2 - \left(r_n^m - D_n^m\right) n\right) \delta t, \\
\beta_n &= 1 - \left(r_n^m + \left(\sigma_n^m\right)^2 n^2\right) \delta t, \\
\gamma_n &= \frac{1}{2}\left(\left(\sigma_n^m\right)^2 n^2 + \left(r_n^m - D_n^m\right) n\right) \delta t
\end{aligned}$$

So the problem of variable drift/diffusion has been trivially implemented.

6 The Greeks

Having discussed finite difference techniques for approximating derivatives, obtaining the greeks becomes a simple task of defining *theta*, *delta* and *gamma*:

$$\begin{aligned} \theta_n^m &\sim \frac{V_n^m - V_n^{m-1}}{\delta t}, \quad \Delta_n^m \sim \frac{V_{n+1}^m - V_{n-1}^m}{2\delta S}, \\ \Gamma_n^m &\sim \frac{V_{n-1}^m - 2V_n^m + V_{n+1}^m}{\delta S^2}. \end{aligned}$$

These are the simple greeks which involve derivatives with respect to a variable. For more advanced greeks (based upon parameters), such as *vega* and *rho,* these can be calculated using FDM by expressing modifications of the BSE. To illustrate this idea let us introduce a convenient form of shorthand by defining the options *vega* υ as

$$\upsilon(S,t) = \frac{\partial V}{\partial \sigma}.$$

Now by differentiating the (earlier) Black-Scholes problem (equation and payoff)

$$\frac{\partial}{\partial \sigma}\left\{\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + (r-D)\, S\frac{\partial V}{\partial S} - rV = 0\right\}$$

and

$$\frac{\partial}{\partial \sigma}\left\{\max\left(S-E,\ 0\right)\right\}$$

the equation becomes

$$\left\{\tfrac{\partial}{\partial t}\frac{\partial V}{\partial \sigma}+\tfrac{1}{2}S^2\tfrac{\partial}{\partial \sigma}\left(\sigma^2\frac{\partial^2 V}{\partial S^2}\right)+\left(r-D\right)S\tfrac{\partial}{\partial S}\frac{\partial V}{\partial \sigma}-r\frac{\partial V}{\partial \sigma}=0\right\}$$

to give the final form

$$\frac{\partial \upsilon}{\partial t}+\frac{1}{2}\sigma^2S^2\frac{\partial^2 \upsilon}{\partial S^2}+\left(r-D\right)S\frac{\partial \upsilon}{\partial S}-r\upsilon=-\sigma S^2\frac{\partial^2 V}{\partial S^2}$$

together with the payoff condition which becomes

$$\frac{\partial V\left(S,T\right)}{\partial \sigma}=\upsilon\left(S,T\right)=0.$$

The resulting PDE is the BSE with a forcing term (on the right hand side) due to the diffusion. Note - in order to obtain this we have assumed the existence of continuous second order partial derivatives so for example

$$\frac{\partial}{\partial \sigma}\frac{\partial V}{\partial t}\equiv\frac{\partial}{\partial t}\frac{\partial V}{\partial \sigma}\rightarrow\frac{\partial \upsilon}{\partial t}.$$

We also note that in arriving at $\upsilon\left(S,T\right)=0$ we have assumed that the payoff is independent of the volatility.

7 Implicit Finite Difference Approximations

We now introduce the Implicit Finite Difference (FD) Scheme.

By using the same notation as in the explicit case we will obtain an implicit difference method for the Black-Scholes equation.

We begin with the Taylor-series approximations for the 1st and 2nd order derivatives, where $V(S,t) = V(n\delta S,\ m\delta t) = V_n^m$:

$$\begin{aligned}
\frac{\partial V}{\partial t} &\sim \frac{V_n^m - V_n^{m-1}}{\delta t} + O(\delta t)\,, \\
\frac{\partial V}{\partial S} &\sim \frac{V_{n+1}^{m-1} - V_{n-1}^{m-1}}{2\delta S} + O\left(\delta S^2\right)\,, \\
\frac{\partial^2 V}{\partial S^2} &\sim \frac{V_{n-1}^{m-1} - 2V_n^{m-1} + V_{n+1}^{m-1}}{\delta S^2} + O\left(\delta S^2\right)
\end{aligned}$$

Substituting in the BSE gives and rearranging to obtain another *backward marching* scheme in time gives the linear system

$$a_n V_{n-1}^{m-1} + b_n V_n^{m-1} + c_n V_{n+1}^{m-1} = V_n^m$$

where

$$\begin{aligned} a_n &= -\frac{1}{2}\left(\sigma^2 n^2 - n\,(r-D)\right)\delta t,\ b_n = 1 + \left(\sigma^2 n^2 + r\right)\delta t, \\ c_n &= -\frac{1}{2}\left(\sigma^2 n^2 + n\,(r-D)\right)\delta t \end{aligned}$$

This expression is accurate to $O\left(\delta S^2,\ \delta t\right)$.

The chief attraction of this method lies in its instability - it is stable for all values of δt and we call this scheme *unconditionally stable.*

As we know V_n^m before V_n^{m-1}, the system is implicit for the V_n^{m-1} term.

Boundary conditions:

At $S = 0$, i.e. $n = 0$, the BSE becomes

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

The problem again arises at $n = N$ $(S \to \infty)$. We cannot use our difference equation at the boundary, as we end up with a term V_{N+1}^{m-1}, which is not defined. So we use the gamma condition mentioned above ($\Gamma \to 0$).

As before we know

$$\Gamma \sim \frac{V_{n-1}^{m-1} - 2V_n^{m-1} + V_{n+1}^{m-1}}{\delta S^2} = 0.$$

Upon rearranging gives at $n = N$

$$V_{N+1}^{m-1} = 2V_N^{m-1} - V_{N-1}^{m-1}$$

and substituting in the difference equation gives

$$\widehat{a}_N V_{N-1}^{m-1} + \widehat{b}_N V_N^{m-1} = V_N^{m-1},$$

where

$$\widehat{a}_N = N\left(r - D\right)\delta t, \quad \widehat{b}_N = 1 - \left(N\left(r - D\right) - r\right)\delta t,$$

In summary, the scheme is:

$$\left.\begin{array}{c} a_n V_{n-1}^{m-1} + b_n V_n^{m-1} + c_n V_{n+1}^{m-1} = V_n^m \\ M \geq m \geq 1; \quad 1 \leq n \leq N-1 \end{array}\right\} \text{ D.E}$$

$$\left.\begin{array}{c} V_n^M = \max\left(n\delta S - E, 0\right) \\ 0 \leq n \leq N; \end{array}\right\} \text{ Final Payoff Condition}$$

$$\left.\begin{array}{c} \left(1 + r\delta t\right) V_0^{m-1} = V_0^m \\ M \geq m \geq 1 \end{array}\right\} \text{Boundary condition at } \quad (S = 0)$$

$$\left.\begin{array}{c} \widehat{a}_N V_{N-1}^{m-1} + \widehat{b}_N V_N^{m-1} = V_N^m \\ M \geq m \geq 1; \quad S = N\delta S \end{array}\right\} \text{Boundary condition at } \; S^*$$

We can write the problem as a system of linear equations, called a *linear system*,

$$\begin{array}{rcl} a_1 V_0^{m-1} + b_1 V_1^{m-1} + c_1 V_2^{m-1} & = & V_1^m \\ a_2 V_1^{m-1} + b_2 V_2^{m-1} + c_2 V_3^{m-1} & = & V_2^m \\ & & \vdots \\ & & \vdots \\ a_{N-1} V_{N-2}^{m-1} + b_{N-1} V_{N-1}^{m-1} + c_{N-1} V_N^{m-1} & = & V_{N-1}^m \end{array}$$

$$\begin{pmatrix} b_0 & c_0 & \cdots & \cdots & \cdots & 0 \\ a_1 & b_1 & c_1 & 0 & & \vdots \\ 0 & a_2 & b_2 & c_2 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & a_{N-1} & b_{N-1} & c_{N-1} \\ 0 & \cdots & \cdots & 0 & \widehat{a}_N & \widehat{b}_N \end{pmatrix} \begin{pmatrix} V_0^{m-1} \\ V_1^{m-1} \\ V_2^{m-1} \\ \vdots \\ V_{N-1}^{m-1} \\ V_N^{m-1} \end{pmatrix} = \begin{pmatrix} V_0^m \\ V_1^m \\ V_2^m \\ \vdots \\ V_{N-1}^m \\ V_N^m \end{pmatrix}$$

So we are solving

$$\mathbf{A}\underline{V}^{m-1} = \underline{V}^m.$$

at each time step for the unknown vector $\underline{V}^{m-1}$. We note that the matrix A is extremely sparse. A matrix consisting of a main diagonal together with one above (*super-diagonal*) and one below (*sub-diagonal*) is called a *tri-diagonal matrix*.

This linear system can now be solved directly or iteratively using e.g. the Gauss-Seidel Method.

8 The Crank-Nicolson Scheme

The fully implicit method has the same order of accuracy (in time and asset price) as the explicit scheme but is unconditionally stable , i.e. $\forall\ \delta t > 0$.

The Crank-Nicolson method whilst being unconditionally stable has the additional advantage that it is also second order accurate in time.

Consider a PDE being satisfied at the midpoint $\left(n\delta S, \left(m - \frac{1}{2}\right)\delta t\right)$ and replace V_S and V_{SS} by means of a FD approximation at the m^{th} and $(m+1)^{\text{th}}$ time steps, i.e.

$$\tfrac{1}{2}\left(m + (m-1)\right) \to \left(m - \tfrac{1}{2}\right).$$

The method is regarded as an **equally** weighted average of the explicit and implicit schemes with advantage over both individual cases due to accuracy being $O\left(\delta t^2\right)$.

We can then write the FD approximations as

$$\begin{aligned}
\frac{\partial V}{\partial t}\left(n\delta S,\left(m-\tfrac{1}{2}\right)\delta t\right) &\sim \frac{V_n^m - V_n^{m-1}}{\delta t}\\
\frac{\partial V}{\partial S}\left(n\delta S,\left(m-\tfrac{1}{2}\right)\delta t\right) &\sim \frac{1}{2}\frac{\partial V}{\partial S}\left(n\delta S, m\delta t\right) + \\
&\quad \frac{1}{2}\frac{\partial V}{\partial S}\left(n\delta S, (m-1)\,\delta t\right)\\
\frac{\partial^2 V}{\partial S^2}\left(n\delta S,\left(m-\tfrac{1}{2}\right)\delta t\right) &\sim \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\left(n\delta S, m\delta t\right) + \\
&\quad \frac{1}{2}\frac{\partial^2 V}{\partial S^2}\left(n\delta S, (m-1)\,\delta t\right)
\end{aligned}$$

and substitute into the BSE as earlier - keeping note of the fact that we are stepping backwards in time. The resulting difference equation is

$$a_n V_{n-1}^{m-1} + b_n V_n^{m-1} + c_n V_{n+1}^{m-1} = A_n V_{n-1}^{m} + B_n V_n^{m} + C_n V_{n+1}^{m}$$

where

$$\begin{aligned}
a_n &= -\frac{1}{4}\Big(\sigma^2 n^2 - n\,(r-D)\Big)\,\delta t \\
b_n &= 1 + \frac{1}{2}\Big(\sigma^2 n^2 + r\Big)\,\delta t \\
c_n &= -\frac{1}{4}\Big(\sigma^2 n^2 + n\,(r-D)\Big)\,\delta t. \\
A_n &= \frac{1}{4}\Big(\sigma^2 n^2 - n\,(r-D)\Big)\,\delta t \\
B_n &= 1 - \frac{1}{2}\Big(\sigma^2 n^2 + r\Big)\,\delta t \\
C_n &= \frac{1}{4}\Big(\sigma^2 n^2 + n\,(r-D)\Big)\,\delta t.
\end{aligned}$$

and the matrix inversion problem we solve is

$$\mathbf{A}\underline{V}^{m-1} = \mathbf{B}\underline{V}^{m}.$$

9 The LU Decomposition

It is often advantageous to think of Gaussian elimination as constructing a lower tridiagonal matrix L and an upper triangular matrix U, so that $LU = A$. To illustrate this method consider the following *tridiagonal* linear system

$$\begin{pmatrix} b_0 & c_0 & 0 & \cdots & \cdots & 0 \\ a_1 & b_1 & c_1 & & & \vdots \\ 0 & a_2 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & a_{n-1} & b_{n-1} & c_{n-1} \\ 0 & \cdots & \cdots & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{pmatrix} \tag{8}$$

We can write this in the form

$$\mathbf{M.y} = \mathbf{p}$$

and then consider the matrix factorisation $\mathbf{M} = \mathbf{LU}$

$$= \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ l_1 & 1 & 0 & & & \vdots \\ 0 & l_2 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & l_{n-1} & 1 & 0 \\ 0 & \cdots & \cdots & 0 & l_n & 1 \end{pmatrix} \begin{pmatrix} d_0 & u_0 & 0 & \cdots & \cdots & 0 \\ 0 & d_1 & u_1 & & & \vdots \\ 0 & 0 & d_2 & u_2 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & 0 & d_{n-1} & u_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & d_n \end{pmatrix}$$

$$= \begin{pmatrix} d_0 & u_0 & 0 & \cdots & \cdots & 0 \\ l_1 d_0 & l_1 u_0 + d_1 & u_1 & & & \vdots \\ 0 & l_2 d_1 & l_2 u_1 + d_2 & u_2 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & l_{n-1} d_{n-2} & l_{n-1} u_{n-2} + d_{n-1} & u_{n-1} \\ 0 & \cdots & \cdots & 0 & l_n d_{n-1} & l_n u_{n-1} + d_n \end{pmatrix}$$

where L is the lower triangular matrix above and U is the upper triangular matrix above.

Equating the elements of the original tridiagonal matrix M with the elements of the product matrix LU, we find that

$$\begin{aligned} d_0 &= b_0 \qquad (9) \\ u_i &= c_i, \quad i = 0, 1, 2, ..., n-1 \\ l_i &= \frac{a_i}{d_{i-1}}, \quad d_i = b_i - l_i u_{i-1}, \quad i = 1, 2, ..., n \end{aligned}$$

We then solve the original system given by (8) by solving two smaller problems. The tridiagonal system (8) is written in the form

$$\mathbf{M}.\mathbf{y} = (\mathbf{LU})\,.\mathbf{y} = \mathbf{L}\,(\mathbf{U}.\mathbf{y}) = \mathbf{p}$$

We introduce an intermediate vector $\mathbf{z} = \mathbf{Uy}$ so that (8) becomes

$$\mathbf{Lz} = \mathbf{p}, \quad \mathbf{Uy} = \mathbf{z}.$$

First we solve the problem

$$\mathbf{Lz} = \mathbf{p}$$

for the intermediate vector $\mathbf{z}$, i.e. we solve the system

$$\begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & 0 \\ l_1 & 1 & 0 & & & \vdots \\ 0 & l_2 & 1 & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & l_{n-1} & 1 & 0 \\ 0 & \cdots & \cdots & 0 & l_n & 1 \end{pmatrix} \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ \vdots \\ z_{n-1} \\ z_n \end{pmatrix} = \begin{pmatrix} p_0 \\ p_1 \\ p_2 \\ \vdots \\ p_{n-1} \\ p_n \end{pmatrix} \qquad (10)$$

Trivially, the z_i can be found by forward substitution so

$$z_0 = p_0, \quad z_i = p_i - l_i z_{i-1}, \quad i = 1, 2, ..., n$$

This determines the vector $\mathbf{z}$.

Having obtained $\mathbf{z}$ we find $\mathbf{y}$ by solving $\mathbf{Uy} = \mathbf{z}$;

$$\begin{pmatrix} d_0 & u_0 & 0 & \cdots & \cdots & 0 \\ 0 & d_1 & u_1 & & & \vdots \\ 0 & 0 & d_2 & u_2 & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & 0 & d_{n-1} & u_{n-1} \\ 0 & \cdots & \cdots & 0 & 0 & d_n \end{pmatrix} \begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ \vdots \\ y_{n-1} \\ y_n \end{pmatrix} = \begin{pmatrix} z_0 \\ z_1 \\ \vdots \\ \vdots \\ z_{n-1} \\ z_n \end{pmatrix} \tag{11}$$

The solution of (11) is trivially obtained by backward substitution

$$y_n = \frac{z_n}{d_n}, \quad y_i = \frac{z_i - u_i y_{i+1}}{d_i}, \quad i = n-1, n-2, ..., 2, 1$$

This gives us the solution, $\mathbf{y}$, of our original problem (8), $\mathbf{My} = \mathbf{p}$.

This method can be also extended to decompose non-sparse matrices, i.e. $A = LU$, thus opening up a wider class of associated methods.

$$= \begin{pmatrix} l_{11} & 0 & \cdots & \cdots & \cdots & 0 \\ l_{21} & l_{22} & 0 & & & \vdots \\ \vdots & l_{32} & & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & l_{n-1} & l_{n-1,n-1} & 0 \\ l_{n1} & l_{n2} & \cdots & \cdots & \cdots & l_{nn} \end{pmatrix} \times$$

$$\begin{pmatrix} u_{11} & u_{12} & \cdots & \cdots & \cdots & u_{1n} \\ 0 & u_{22} & u_{23} & \cdots & \cdots & u_{2n} \\ 0 & 0 & \ddots & \ddots & & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \vdots \\ \vdots & & & 0 & u_{n-1,n-1} & u_{n-1,n} \\ 0 & \cdots & \cdots & 0 & 0 & u_{nn} \end{pmatrix}$$

10 Iterative Techniques

Consider the $(n \times n)$ matrix inversion problem: $A\mathbf{x} = \mathbf{b}$

Set the initial approximation $\mathbf{x}^{(0)}$ to the solution $\mathbf{x}$, which generates a sequence of vectors $\left\{\mathbf{x}^{(k)}\right\}_{k=0}^{\infty}$ that converges to $\mathbf{x}$.

Most techniques involve a process which converts $A\mathbf{x} = \mathbf{b}$ to

$$\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$$

for each $k = 1, 2, 3,$ ...., where T is $(n \times n)$ and $\mathbf{c} \in \mathbb{R}^n$ is a vector. The initial vector $\mathbf{x}^{(0)}$ is selected and a sequence of approximations generated by computing the new system above.

As an example consider the following linear system

$$\begin{aligned}
10x_1 - x_2 + 2x_3 \quad &= 6 \\
-x_1 + 11x_2 - x_3 + 3x_4 \quad &= 25 \\
2x_1 - x_2 + 10x_3 - x_4 \quad &= -11 \\
3x_2 - x_3 + 8x_4 \quad &= 15
\end{aligned}$$

which has an exact solution $\mathbf{x} = (1, 2, -1, 1)$.

To convert $A\mathbf{x} = \mathbf{b}$ to the form $\mathbf{x}^{(k)} = T\mathbf{x}^{(k-1)} + \mathbf{c}$; solve for each x_i $(i = 1,\ 2,\ 3,\ 4)$ from

$$\left.\begin{array}{c}
x_1^{(k)} = \frac{1}{10}\left\{x_2^{(k-1)} - 2x_3^{(k-1)}\right\} + \frac{3}{5} \\
x_2^{(k)} = \frac{1}{11}\left\{x_1^{(k-1)} + x_3^{(k-1)} - 3x_4^{(k-1)}\right\} + \frac{25}{11} \\
x_3^{(k)} = \frac{1}{10}\left\{-2x_1^{(k-1)} + x_2^{(k-1)} + x_4^{(k-1)}\right\} - \frac{11}{10} \\
x_4^{(k)} = \frac{1}{8}\left\{-3x_2^{(k-1)} + x_3^{(k-1)}\right\} + \frac{15}{8}
\end{array}\right\} \tag{12}$$

$\Rightarrow$

$$T = \begin{bmatrix} 0 & \frac{1}{10} & -\frac{1}{5} & 0 \\ \frac{1}{11} & 0 & \frac{1}{11} & -\frac{3}{11} \\ -\frac{1}{5} & \frac{1}{10} & 0 & \frac{1}{10} \\ 0 & -\frac{3}{8} & \frac{1}{8} & 0 \end{bmatrix}, \quad \mathbf{c} = \begin{bmatrix} \frac{3}{5} \\ \frac{25}{11} \\ -\frac{11}{10} \\ \frac{15}{8} \end{bmatrix}$$

For an initial approximation let $\mathbf{x}^{(0)} = \mathbf{0}^{\mathsf{T}}$ and generate $\mathbf{x}^{(1)}$ by substituting $(0,\ 0,\ 0,\ 0)^{\mathsf{T}}$ in (12) to get

$$(0.6,\ 2.2727, -1.1000,\ 1.8750) = \mathbf{x}^{(1)}$$

and $\mathbf{x}^{(2)}$ can now be obtained.

In general obtain $\mathbf{x}^{(k)} = \left(x_1^{(k)},\ x_2^{(k)},\ x_3^{(k)},\ x_4^{(k)}\right)^{\mathsf{T}}$.

By the very nature of iterative techniques, these continue indefinitely, hence one requires a convergence criterion to terminate the computation once the imposed condition is satisfied. So we need to know when to stop iterating, i.e. does the sequence converge to the solution of the given system of equations? To answer this, we need a means of measuring the distance between $n-$ dimensional vectors.

The most appropriate form for our convergence criteria makes use of the l_∞ norm together with the condition

$$\frac{\left\|\mathbf{x}^{(k)}-\mathbf{x}^{(k-1)}\right\|_\infty}{\left\|\mathbf{x}^{(k-1)}\right\|_\infty} < \varepsilon.$$

ε is the specified tolerance (>0) and $\|\mathbf{x}\|_\infty = \max\limits_{1\leq\, i\,\leq\, n} |x_i|$.

In the numerical example, actual computations yield for $k = 9$ & 10

$k = 9$	$k = 10$
0.9997	1.0001
2.0004	1.9998
-1.0004	-0.9998
1.0006	0.9998

based on $\varepsilon = 10^{-3}$

and the convergence condition yields

$$\frac{\left\|\mathbf{x}^{(10)}-\mathbf{x}^{(9)}\right\|_\infty}{\left\|\mathbf{x}^{(9)}\right\|_\infty} = \frac{\left\|(4,\ -6,\ 6,\ -8)\,10^{-4}\right\|_\infty}{2.0004} \approx 4\times 10^{-4} < \varepsilon$$

Also $\left\|\mathbf{x}^{(10)}-\mathbf{x}^{(9)}\right\|_\infty = 2\times 10^{-4} < \varepsilon$

The iterative technique used is called the *JACOBI* method.

It consists of solving the i^{th} equation in $A\mathbf{x}=\mathbf{b}$ for x_i (provided $a_{ii}\neq 0$) to obtain

$$x_i = \sum_{j=1}^{n}\left(\frac{-a_{i\ j}x_j}{a_{ii}}\right)+\frac{b_i}{a_{ii}} \qquad i=1,\dots\dots,n \quad ; \quad j\neq i$$

and generating each $x_i^{(k)}$ from components of $\mathbf{x}^{(k-1)}$ $(k\geq 1)$ by

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[\sum_{j=1}^{n} \left(-a_{i\ j}\ x_j^{(k-1)} \right) + b_i \right] \ ,\ i = 1, \ldots.., n;\ j \neq i$$

We can now refine this method very simply by using the most up-to-date x_i.

To compute $x_i^{(k)}$, components of $\underline{x}^{(k-1)}$ are used.

Since for $i > 1$, $x_1^{(k)}$, ..., $x_{i-1}^{(k)}$ have already been computed and are likely to be better approximations to the actual solutions x_1, ..., x_{i-1} than $x_1^{(k-1)}$, ..., $x_{i-1}^{(k-1)}$, it is far better to calculate $x_i^{(k)}$ using the most recently calculated values, i.e.

$$x_i^{(k)} = \frac{1}{a_{ii}} \left[-\sum_{j=1}^{i-1} \left(a_{i\ j}\ x_j^{(k)} \right) - \sum_{j=i+1}^{n} \left(a_{i\ j}\ x_j^{(k-1)} \right) + b_i \right]$$
$$(i = 1, \ldots\ldots., n)$$

This is called the *GAUSS-SEIDEL* method.

Re-doing the previous example using the G-S method iterative technique:

$$10x_1 - x_2 + 2x_3 = 6$$

$$-x_1 + 11x_2 - x_3 + 3x_4 = 25$$

$$2x_1 - x_2 + 10x_3 - x_4 = -11$$

$$3x_2 - x_3 + 8x_4 = 15$$

which is written as

$$\left.\begin{array}{c}
x_1^{(k)} = \frac{1}{10}\left\{x_2^{(k-1)} - 2x_3^{(k-1)} + 6\right\} \\
x_2^{(k)} = \frac{1}{11}\left\{x_1^{(k)} + x_3^{(k-1)} - 3x_4^{(k-1)} + 25\right\} \\
x_3^{(k)} = \frac{1}{10}\left\{-2x_1^{(k)} + x_2^{(k)} + x_4^{(k-1)} - 11\right\} \\
x_4^{(k)} = \frac{1}{8}\left\{-3x_2^{(k)} + x_3^{(k)} + 15\right\}
\end{array}\right\}$$

Again letting $\mathbf{x}^{(0)} = \underline{0}^{\mathsf{T}}$

Here

$\mathbf{x}^{(4)} = (1.0009,\ 2.0003,\ -1.0003,\ 0.9999)$

$\mathbf{x}^{(5)} = (1.0001,\ 2.0000,\ -1.0000,\ 1.0000)$

$$\frac{\left\|\mathbf{x}^{(5)} - \mathbf{x}^{(4)}\right\|_\infty}{\left\|\mathbf{x}^{(4)}\right\|_\infty} \approx 4 \times 10^{-4} \quad (< \varepsilon)$$

We note that Jacobi's method required twice as many iterations for the same level of accuracy.

If A is strictly diagonally dominant than for any choice of $\mathbf{x}^{(0)}$ the sequence of solutions generated by both Gauss-Seidel and Jacobi converge to the unique solution.

10.1 Successive-Over-Relaxation (SOR) Methods

A large part of numerical analysis is based upon the need to improve the efficiency and speed of existing techniques. Although we have seen from the Gauss-Seidel method that it is superior to Jacobi in this regard, we can nevertheless continue to look for improvements in convergence speed. In this section we will briefly look at a method called *successive-over-relaxation*.

Let $\underline{\widehat{x}} \in \mathbb{R}^n$ be an approximation to the actual solution for $A\underline{x} = \underline{b}$.

The *residual vector* $\underline{r}$ for $\underline{\widehat{x}}$ is

$$\underline{r} = \underline{b} - A\underline{\widehat{x}}$$

In Jacobi/Gauss-Seidel methods a residual vector is associated with each calculation of an approximation component to the solution vector.

Aim: Generate a sequence of approximations that cause the associated residual vectors to converge rapidly to zero. Then methods involving equations of the form

$$x_i^{(k)} = x_i^{(k-1)} + \omega \frac{r_{ii}^{(k)}}{a_{ii}}$$

are called **relaxation methods** and for certain choices of positive ω lead to faster convergence.

For $0 < \omega < 1$, the procedures are called **under-relaxation methods** and used for convergence of systems which do not converge by Gauss-Seidel.

For $\omega > 1$, the procedures are called **over-relaxation methods** which are used to accelerate convergence of systems which are convergent by Gauss-Seidel.

Scheme: For Gauss-Seidel we have

$$\begin{aligned} x_i^{(k)} &= x_i^{(k-1)}(1-\omega) + \\ & \frac{\omega}{a_{ii}} \left[b_i - \sum_{j=1}^{i-1} \left(a_{i\ j}\ x_j^{(k)} \right) - \sum_{j=i+1}^{n} \left(a_{i\ j}\ x_j^{(k-1)} \right) \right] \end{aligned}$$

for $(i = 1, \ldots\ldots, n)$.

11 The $\theta-$Method

This is a method that manages to have a local truncation error of $O(\delta S^4, \delta t^2)$ for the same computational effort as the Crank–Nicolson scheme.

The basic diffusion equation is

$$\frac{\partial V}{\partial t} + \frac{\partial^2 V}{\partial S^2} = 0.$$

The explicit method applied to this equation is just

$$\frac{V_n^{m+1} - V_n^m}{\delta t} = \frac{V_{n+1}^m - 2V_n^m + V_{n-1}^m}{\delta S^2}.$$

and the fully implicit is similarly

$$\frac{V_n^{m+1} - V_n^m}{\delta t} = \frac{V_{n+1}^{m+1} - 2V_n^{m+1} + V_{n-1}^{m+1}}{\delta S^2}.$$

The Crank–Nicolson scheme is an average of these two methods. What about a *weighted* average?

This leads to the θ method.

Take a weighted average of the explicit and implicit methods to get

$$\begin{aligned}\frac{V_n^{m+1}-V_n^m}{\delta t} &= \theta\left(\frac{V_{n+1}^{m+1}-2V_n^{m+1}+V_{n-1}^{m+1}}{\delta S^2}\right)\\ &\quad +(1-\theta)\left(\frac{V_{n+1}^{m}-2V_n^{m}+V_{n-1}^{m}}{\delta S^2}\right).\end{aligned}$$

When $\theta = 1/2$ we are back to the Crank–Nicolson method.

For a general value of θ the local truncation error is

$$O\left(\frac{1}{2}\delta t + \frac{1}{12}\delta S^2 - \theta\delta t, \delta S^4, \delta t^2\right).$$

When $\theta = 0$, $1/2$ or 1 we get the results we have seen so far.

But if

$$\theta = \frac{1}{2} - \frac{\delta S^2}{12\delta t}$$

then the local truncation error is improved.

The implementation of the method is no harder than the Crank–Nicolson scheme.

12 Three time-level methods

Numerical schemes are not restricted to the use of just two time levels. So far we have considered $\left(V^{m+1}, V^m\right)$ or $\left(V^m, V^{m-1}\right)$

We can construct many algorithms using three or more time levels, i.e.

$$V^{m+1} = g\left(V^{m-1}, V^m\right)$$

Again, we would do this if it gave us a better local truncation error or had better convergence properties. We already know that the centred time difference

$$\frac{V_n^{m+1} - V_n^{m-1}}{2\delta t}$$

is unstable for all values of δt. However the scheme

$$\frac{\left(V_n^{m+1} - V_n^{m-1}\right)}{2\delta t} = \frac{1}{\delta S^2}\left(V_{n+1}^m \underbrace{- V_n^{m+1} - V_n^{m-1}}_{=-2V_n^m} + V_{n-1}^m\right)$$

is stable for all δt.

13 Jump conditions

As well as having final conditions (the payoff), boundary conditions (at zero, infinity or at a barrier) and the PDE to satisfy we often have jump conditions.

These can be due to a jump in the underlying on a dividend date, the payment of some coupon, or because of the jump in a discretely-sampled path-dependent quantity.

13.1 A discrete cashflow

If our contract entitles us to a discretely-paid sum on a specified date, then the contract value must jump by the amount of the cashflow. In continuous time we have

$$V(S, t_d^-) = V(S, t_d^+) + C.$$

The cashflow C may be a function of the underlying, or even of the option value.

From a numerical point of view we would ideally like the date t_d on which there is a cashflow to lie exactly on a grid point. If this is the case then the implementation of the above condition is straightforward. Simply calculate the option value up to and including the date t_d, then before moving on to the time step after the payment just add C to every option value.

Complications arise if the date does not coincide with grid points.

The simplest implementation is to just add C at the first time step after the date.

This is accurate to $O(\delta t)$ and so is consistent for the explicit method.

If you are using a more accurate numerical method such as Crank–Nicolson you should use a more accurate implementation of the jump condition.

The best thing to do is to make the time step just after the cashflow of the correct size to match up the grid with the cashflow date.

That way the cashflow is guaranteed to be incorporated at the same order of accuracy as the numerical method itself.

14 Path-dependent options

Many path-dependent contracts have simple PDE representations.

Often the problem is in three dimensions, the usual underlying (asset, interest rate etc.), time and the path-dependent quantity.

These three-dimensional problems for exotic options come in two forms:

- in one there are no new terms in the equation because the path-dependent quantity is measured discretely

- in the other there is an extra first derivative with respect to the new continuously-sampled variable

Clearly the most important aspect of the path-dependent problem is the extra dimension. Thus far the option price has been a 2D problem, i.e. $V = V(S,t)$. We must now solve in three dimensions, due to the sampling variable I and must therefore introduce the option value as

$$V(S,t,I) = V_{n,j}^{m}.$$

The superscript m still refers to time, the subscript n to the asset but now j refers to the new variable.

14.1 Continuously sampled quantities

When the path-dependent quantity is sampled continuously there is usually a new term in the PDE and thus new terms in a difference equation approximation. The resulting PDE is of the form:

$$\frac{\partial V}{\partial t}+a(S,I,t)\frac{\partial^2 V}{\partial S^2}+b(S,I,t)\frac{\partial V}{\partial S}+f(S,I,t)\frac{\partial V}{\partial I}+c(S,I,t)V=0$$

We are referring in particular to Asian options - where the payoff depends on some average.

The updating of the scheme, the implementation of the boundary and final conditions are all very straightforward.

The only new point is the choice of the discrete version of the derivative

$$\frac{\partial V}{\partial I}.$$

Because there is no diffusion in the I direction the choice of difference is important.

The derivative with respect to I represents convection in the I direction and the numerical scheme must be consistent with this.

- For this reason a one-sided difference must be used

If the coefficient $f(S, I, t)$ changes sign then the choice of difference must reflect this, upwind differencing *must* be used.

Here is a possible choice for the difference:

$$\begin{aligned} \text{if } f(S, I, t) &\geq 0 \text{ then} \\ f(S, I, t)\frac{\partial V}{\partial I}(S, I, t) &= f^m_{n,j+\frac{1}{2}}\frac{V^m_{n,j+1} - V^m_{n,j}}{\delta I} \end{aligned}$$

but if

$$\begin{aligned} \text{if } f(S,I,t) &< 0 \text{ then} \\ f(S,I,t)\frac{\partial V}{\partial I}(S,I,t) &= f^m_{n,j-\frac{1}{2}}\frac{V^m_{n,j}-V^m_{n,j-1}}{\delta I}. \end{aligned}$$

15 Two Factor Models

Many currently popular financial models have two random factors.

Convertible bonds are usually priced with both random underlying and random interest rate.

Exotic equity derivatives are often priced with stochastic volatility, this is especially true of barrier options.

Finite-difference methods are quite suited to such problems.

- Finite-difference methods work well in low dimensions, and handle early exercise very efficiently.

  Some path dependency is also easy to cope with.

We are going to refer to the general two-factor equation

$$\frac{\partial V}{\partial t} + a(S,r,t)\frac{\partial^2 V}{\partial S^2} + b(S,r,t)\frac{\partial V}{\partial S} + c(S,r,t)V$$

$$+d(S,r,t)\frac{\partial^2 V}{\partial r^2} + e(S,r,t)\frac{\partial^2 V}{\partial S \partial r} + f(S,r,t)\frac{\partial V}{\partial r} = 0. \tag{13}$$

It will be quite helpful if we think of solving the two-factor convertible bond problem.

For the general two-factor problem (13) to be parabolic we need

$$e(S,r,t)^2 < 4a(S,r,t)\,d(S,r,t).$$

As in the one-factor world, the variables must be discretized.

That is, we solve on a three-dimensional grid with

$$S = n\delta S, \quad r = j\delta r, \text{ and } t = T - m\delta t.$$

Expiry is $t = T$ or $m = 0$.

The indices range from zero to N and J for n and j respectively, and from zero to M for m.

We will assume that the interest rate model is only specified on $r \geq 0$.

This may not be the case, some simple interest rate models such as Vasicek, are defined over negative r as well.

The contract value is written as

$$V(S, r, t) = V_{nj}^{m}.$$

Whatever the problem to be solved, we must impose certain conditions on the solution.

First of all, we must specify the final condition.

This is the payoff function, telling us the value of the contract at the expiration of the contract.

Suppose that we are pricing a long-dated warrant with a call payoff.

The final condition for this problem is then

$$V(S, r, T) = V_{nj}^{0} = \mathsf{max}(S - E, 0).$$

Boundary conditions must be imposed at all the grid points marked with a dot. The boundary conditions will depend on the contract.

15.1 The explicit method

The one-factor explicit method can be extended to two-factors with very little effort.

In fact, the ease of programming make it a very good method for those new to the subject.

We will use symmetric central differences for all derivatives in (13).

This is the best way to approximate the second derivatives but may not be the best for the first derivatives.

We have seen how to use central differences for all of the terms with the exception of the second derivative with respect to both S and r,

$$\frac{\partial^2 V}{\partial S \partial r}.$$

We can approximate this by

$$\frac{\partial\left(\frac{\partial V}{\partial r}\right)}{\partial S} \approx \frac{\frac{\partial V}{\partial r}(S+\delta S, r, t) - \frac{\partial V}{\partial r}(S-\delta S, r, t)}{2\,\delta S}.$$

But

$$\frac{\partial V}{\partial r}(S+\delta S, r, t) \approx \frac{V^m_{n+1,j+1} - V^m_{n+1,j-1}}{2\,\delta r}.$$

This suggests that a suitable discretization might be

$$\frac{\frac{V^m_{n+1,j+1}-V^m_{n+1,j-1}}{2\,\delta r} - \frac{V^m_{n-1,j+1}-V^m_{n-1,j-1}}{2\,\delta r}}{2\,\delta S}$$

$$= \frac{V^m_{n+1,j+1} - V^m_{n+1,j-1} - V^m_{n-1,j+1} + V^m_{n-1,j-1}}{4\delta S \delta r}.$$

This is particularly good since, not only is the error of the same error as in the other derivative approximations but also it preserves the property that

$$\frac{\partial^2 V}{\partial S \partial r} = \frac{\partial^2 V}{\partial r \partial S}.$$

The resulting explicit difference scheme is

$$\frac{V_{nj}^{m} - V_{nj}^{m+1}}{\delta t} + a_{nj}^{k}\left(\frac{V_{n+1,j}^{m} - 2V_{nj}^{m} + V_{n-1,j}^{m}}{\delta S^2}\right)$$

$$+b_{nj}^{m}\left(\frac{V_{n+1,j}^{m} - V_{n-1,j}^{m}}{2\,\delta S}\right) + c_{nj}^{m}V_{nj}^{m}$$

$$+d_{nj}^{m}\left(\frac{V_{n,j+1}^{m} - 2V_{nj}^{m} + V_{n,j-1}^{m}}{\delta r^2}\right)$$

$$e_{nj}^{m}\left(\frac{V_{n+1,j+1}^{m} - V_{n+1,j-1}^{m} - V_{n-1,j+1}^{m} + V_{n-1,j-1}^{m}}{4\,\delta S\,\delta r}\right)$$

$$+f_{nj}^{m}\left(\frac{V_{n,j+1}^{m} - V_{n,j-1}^{m}}{2\,\delta r}\right) = O(\delta t, \delta S^2, \delta r^2).$$

We could rewrite this in the form

$$V_{nj}^{m+1} = \dots,$$

where the right-hand side is a linear function of nine option values, whose coefficients at time step m are related to a, b etc.

It would not be very helpful to write the difference equation in this form, since the actual implementation is usually more transparent than this.

Note that in general all nine points (n, j), $(n \pm 1, j)$, $(n, j \pm 1)$, $(n \pm 1, j \pm 1)$ are used in the scheme.

If there is no cross derivative term then only the five points (n, j), $(n \pm 1, j \pm 1)$ are used.

This simplifies some of the methods.

15.1.1 Stability of the explicit method

One of the advantages of the explicit method is again that it is easy to program. The main disadvantage is that the method is only stable for sufficiently small time steps.

We can analyse stability by looking for solutions of the difference equation that are oscillatory in both the S and r directions:

$$V_{nj}^{m} = \alpha^{m} e^{2i\pi\left(\frac{n}{\lambda_S}+\frac{j}{\lambda_r}\right)}; \quad i = \sqrt{-1}$$

If we have a pure diffusion problem with no correlation between the variables, then the stability requirement becomes

$$a\frac{\delta t}{\delta S^2} + d\frac{\delta t}{\delta r^2} \le \frac{1}{2}.$$

To overcome this time step constraint we can try an implicit scheme.

15.2 Alternating Direction Implicit

We now discuss a type of implicit finite-difference methods used for two-factor problems.

The technique is called **Alternating Direction Implicit** or **ADI**.

We could try a two-factor extension of Crank–Nicolson to solve $(N-1)(J-1)$ equations in the same number of unknowns.

This suffers from the problem that the resulting matrix does not have a nice form, and the solution is complicated and time consuming.

If we want to keep the stability advantage of the implicit method and the ease of solution of the explicit method we could try to solve implicitly in one factor but explicitly in the other.

This is the idea behind ADI.

As well as V_{nj}^m, introduce an 'intermediate' value $V_{nj}^{m+1/2}$.

Solve from time step m to the intermediate step $m+1/2$ using explicit differences in S and implicit differences in r.

Since only one direction is implicit, the solution by LUD decomposition or SORA is no harder than in one factor.

Having found the intermediate value $V_{nj}^{m+1/2}$ step forward to time step $m+1$ using implicit differences in S and explicit differences in r.

For this half time step we have changed around explicit and implicit from the previous half time step.

Again the matrix equations are straightforward.

The method is stable for all time steps and the error is $O(\delta t^2, \delta S^2, \delta r^2)$.