Numerical Methods in Finance II

Lecture 1 - Finite Difference Methods

T.A. McWalter

M.Phil. in Financial Mathematics

2020

Course Outline

- This course is a little less structured than the previous Numerical Methods course given in the first semester.
- In particular, we shall cover a number of disjoint topics. The emphasis is on exposing you to a "toolbox" of techniques that are of practical use.
- We cover the following topics
  - Finite difference methods;
  - Pricing American options;
  - Fourier transform and cosine methods for pricing derivatives;
  - Counterparty credit risk modelling;
  - Stochastic risk-free rates, using both short rate and forward rate/market model approaches;
  - Local and stochastic volatility models.
- Lecture notes will consist of the slides used during the lectures.
- I will, however, treat this more like a "reading course." This means that the notes will be augmented by chapters from books or academic papers.
- The practical tutorials follow the usual format.

Introduction

- Finite difference methods allow simple and powerful numerical solutions for partial differential equations.
- If applied correctly, they are relatively simple and accurate.
- We initially concentrate our efforts on finding numerical solutions to the diffusion equation — this is due to the fact that the Black-Scholes equation may be rewritten in this format, using a suitable transformation of variables.
- In turn, we look at the explicit, implicit, Crank-Nicolson and theta finite-difference schemes.
- Since these methods are approximations, the idea of stability is addressed.
- Using a transformation of the Black-Scholes PDE and writing the initial and boundary conditions in a convenient format, we implement a generic European option pricer.

Classification of Second Order PDEs

- The *order* of a PDE for the function $u(x_1, x_2, \ldots, x_n)$ is the largest number of consecutive applications of a partial derivative with respect to one variable applied to u in the PDE.
- In this module, we will restrict ourselves to functions of two variables $(x, \tau) \in \mathbb{R} \times \mathbb{R}^+$ and to PDEs of second order.
- All second order PDEs for the function $u(x, \tau)$ may be written in the general form

$$a\frac{\partial^2 u}{\partial \tau^2} + b\frac{\partial^2 u}{\partial \tau \partial x} + c\frac{\partial^2 u}{\partial x^2} + d\frac{\partial u}{\partial \tau} + e\frac{\partial u}{\partial x} + fu + g = 0,$$

  where a, b, c, d, e, f and g are functions of τ and x. We classify the PDE as elliptic, parabolic or hyperbolic depending on the sign of the discriminant $b^2 - 4ac$:
  - If $b^2 - 4ac < 0$, the equation is said to be *elliptic*;
  - If $b^2 - 4ac = 0$, the equation is said to be *parabolic*; and
  - If $b^2 - 4ac > 0$, the equation is said to be *hyperbolic*.
- The diffusion equation is an example of a parabolic PDE.
- We shall, therefore, concentrate on numerical methods for PDEs of parabolic type.

Theorem 1.1 (Taylor's theorem) *If $f : [\alpha, \beta] \to \mathbb{R}$ is n times differentiable on $[\alpha, \beta]$, then, for $a, b \in [\alpha, \beta]$, there exists some c between a and b such that*

$$f(a) = \sum_{i=0}^{n-1} \frac{f^{(i)}(b)}{i!}(a-b)^i + \frac{f^{(n)}(c)}{n!}(a-b)^n,$$

where $f^{(i)}$ is the ith derivative with respect to the independent variable. The function

$$R_n(a) = f(a) - \sum_{j=0}^{n-1} \frac{f^{(j)}(b)}{j!}(a-b)^j,$$

is called the nth remainder of f at b.

- Suppose $u : \Omega \to \mathbb{R}$ (where $\Omega \subseteq \mathbb{R} \times \mathbb{R}^+$) is a function of x and τ. Assuming that u is sufficiently smooth, we may use Taylor's formula to derive a number of finite-difference approximations for the partial derivatives of u over the domain Ω.

- We start by approximating $\partial u/\partial \tau$.

Let $(x, \tau) \in \Omega$ and suppose that $\delta_\tau > 0$ is small enough so that $(x, \tau \pm \delta_\tau) \in \Omega$. Then, Taylor expansions of u about (x, τ) yield

$$u(x, \tau - \delta_\tau) = u(x, \tau) - \delta_\tau \frac{\partial u}{\partial \tau}(x, \tau) + \frac{1}{2}\delta_\tau^2 \frac{\partial^2 u}{\partial \tau^2}(x, \tau^-), \tag{1}$$

$$u(x, \tau + \delta_\tau) = u(x, \tau) + \delta_\tau \frac{\partial u}{\partial \tau}(x, \tau) + \frac{1}{2}\delta_\tau^2 \frac{\partial^2 u}{\partial \tau^2}(x, \tau^+), \tag{2}$$

for some τ^-, τ^+ satisfying $\tau - \delta_\tau < \tau^- < \tau < \tau^+ < \tau + \delta_\tau$.

Using (1) leads to the *backward difference* approximation for $\partial u/\partial \tau$

$$\begin{aligned}\frac{\partial u}{\partial \tau}(x, \tau) &= \frac{u(x, \tau) - u(x, \tau - \delta_\tau)}{\delta_\tau} + \frac{1}{2}\delta_\tau \frac{\partial^2 u}{\partial t^2}(x, \tau^-) \\ &= \frac{u(x, \tau) - u(x, \tau - \delta_\tau)}{\delta_\tau} + O(\delta_\tau),\end{aligned}$$

while using (2) leads to the *forward difference approximation*

$$\frac{\partial u}{\partial \tau}(x, \tau) = \frac{u(x, \tau + \delta_\tau) - u(x, \tau)}{\delta_\tau} + O(\delta_\tau).$$

Expanding (1) and (2) further gives

$$u(x,\tau-\delta_\tau) = u(x,\tau) - \delta_\tau \frac{\partial u}{\partial \tau}(x,\tau) + \frac{1}{2}\delta_\tau^2 \frac{\partial^2 u}{\partial \tau^2}(x,\tau) - \frac{1}{6}\delta_\tau^3 \frac{\partial^3 u}{\partial \tau^3}(x,\bar{\tau}^+),$$
$$u(x,\tau+\delta_\tau) = u(x,\tau) + \delta_\tau \frac{\partial u}{\partial \tau}(x,\tau) + \frac{1}{2}\delta_\tau^2 \frac{\partial^2 u}{\partial \tau^2}(x,\tau) + \frac{1}{6}\delta_\tau^3 \frac{\partial^3 u}{\partial \tau^3}(x,\bar{\tau}^-),$$

for some $\bar{\tau}^-$, $\bar{\tau}^+$ satisfying $\tau - \delta_\tau < \bar{\tau}^- < \tau < \bar{\tau}^+ < \tau + \delta_\tau$.

Subtracting the first of these equations from the other produces the central difference approximation

$$\begin{aligned}\frac{\partial u}{\partial \tau}(x,\tau) &= \frac{u(x,\tau+\delta_\tau) - u(x,\tau-\delta_\tau)}{2\delta_\tau} - \frac{1}{6}\delta_\tau^2 \left(\frac{\partial^3 u}{\partial \tau^3}(x,\bar{\tau}^+) + \frac{\partial^3 u}{\partial \tau^3}(x,\bar{\tau}^-) \right) \\ &= \frac{u(x,\tau+\delta_\tau) - u(x,\tau-\delta_\tau)}{2\delta_\tau} + O\left(\delta_\tau^2\right).\end{aligned}$$

- We now turn our attention to approximating the double (spatial) derivative $\partial^2 u/\partial x^2$.

Suppose that $\delta_x > 0$ is small enough so that $(x \pm \delta_x, \tau) \in \Omega$, then Taylor expansions of u about (x,τ) yield

$$\begin{aligned}u(x-\delta_x,\tau) = u(x,\tau) &- \delta_x \frac{\partial u}{\partial x}(x,\tau) + \frac{1}{2}\delta_x^2 \frac{\partial^2 u}{\partial x^2}(x,\tau) \\ &- \frac{1}{6}\delta_x^3 \frac{\partial^3 u}{\partial x^3}(x,\tau) + \frac{1}{24}\delta_x^4 \frac{\partial^4 u}{\partial x^4}(x^-,\tau), \\ u(x+\delta_x,\tau) = u(x,\tau) &+ \delta_x \frac{\partial u}{\partial x}(x,\tau) + \frac{1}{2}\delta_x^2 \frac{\partial^2 u}{\partial x^2}(x,\tau) \\ &+ \frac{1}{6}\delta_x^3 \frac{\partial^3 u}{\partial x^3}(x,\tau) + \frac{1}{24}\delta_x^4 \frac{\partial^4 u}{\partial x^4}(x^+,\tau),\end{aligned}$$

for some x^-, x^+ satisfying $x - \delta_x < x^- < x < x^+ < x + \delta_x$.

Adding these two equations together gives the symmetric central difference approximation for $\partial^2 u/\partial x^2$

$$\begin{aligned}\frac{\partial^2 u}{\partial x^2}(x,\tau) &= \frac{u(x-\delta_x,\tau) - 2u(x,\tau) + u(x+\delta_x,\tau)}{\delta_x^2} \\ &\qquad - \frac{1}{12}\delta_x^2 \left(\frac{\partial^4 u}{\partial x^4}(x^-,\tau) + \frac{\partial^4 u}{\partial x^4}(x^+,\tau) \right) \\ &= \frac{u(x-\delta_x,\tau) - 2u(x,\tau) + u(x+\delta_x,\tau)}{\delta_x^2} + O\left(\delta_x^2\right).\end{aligned}$$

The Finite-difference Mesh

- To make use of the finite-difference approximations that we have derived, we must now consider discretising the problem space.
- As mentioned earlier, we consider one space dimension and time.
- We divide the space axis into equally spaced nodes a distance δ_x apart, and do the same for the time axis using a spacing of δ_τ.
- We are then only concerned with the values of $u(x,\tau)$ at the mesh points $(n\delta_x, m\delta_\tau)$ for some $m, n \in \mathbb{Z}$.
- As a convenience we use the shorthand notation

$$u_m^n = u(n\delta_x, m\delta_\tau).$$

- See Figure 8.2 on page 139 of the notes (WHD).

Exercise

There are a large number of other finite difference formulae; see if you can derive the following difference formulae (assume u is suitably smooth):

1. Forward difference approximations with error $O(\delta_x)$.
   (a) $\dfrac{\partial u}{\partial x}(n\delta_x, m\delta_\tau) \approx \dfrac{u_m^{n+1} - u_m^n}{\delta_x}$.
   (b) $\dfrac{\partial^2 u}{\partial x^2}(n\delta_x, m\delta_\tau) \approx \dfrac{u_m^{n+2} - 2u_m^{n+1} + u_m^n}{\delta_x^2}$.
   (c) $\dfrac{\partial^3 u}{\partial x^3}(n\delta_x, m\delta_\tau) \approx \dfrac{u_m^{n+3} - 3u_m^{n+2} + 3u_m^{n+1} - u_m^n}{\delta_x^3}$.
2. Forward difference approximations with error $O(\delta_x^2)$.
   (a) $\dfrac{\partial u}{\partial x}(n\delta_x, m\delta_\tau) \approx \dfrac{-u_m^{n+2} + 4u_m^{n+1} - 3u_m^n}{2\delta_x}$.
   (b) $\dfrac{\partial^2 u}{\partial x^2}(n\delta_x, m\delta_\tau) \approx \dfrac{-u_m^{n+3} + 4u_m^{n+2} - 5u_m^{n+1} + 2u_m^n}{\delta_x^2}$.
   (c) $\dfrac{\partial^3 u}{\partial x^3}(n\delta_x, m\delta_\tau) \approx \dfrac{-3u_m^{n+4} + 14u_m^{n+3} - 24u_m^{n+2} + 18u_m^{n+1} - 5u_m^n}{2\delta_x^3}$.
3. Backward difference approximations with error $O(\delta_x)$.
   (a) $\dfrac{\partial u}{\partial x}(n\delta_x, m\delta_\tau) \approx \dfrac{u_m^n - u_m^{n-1}}{\delta_x}$.
   (b) $\dfrac{\partial^2 u}{\partial x^2}(n\delta_x, m\delta_\tau) \approx \dfrac{u_m^{n-2} - 2u_m^{n-1} + u_m^n}{\delta_x^2}$.
   (c) $\dfrac{\partial^3 u}{\partial x^3}(n\delta_x, m\delta_\tau) \approx \dfrac{-u_m^{n-3} + 3u_m^{n-2} - 3u_m^{n-1} + u_m^n}{\delta_x^3}$.

4. Backward difference approximations with error $O(\delta_x^2)$.

(a) $$\frac{\partial u}{\partial x}(n\delta_x, m\delta_\tau) \approx \frac{u_m^{n-2} - 4u_m^{n-1} + 3u_m^n}{2\delta_x}.$$

(b) $$\frac{\partial^2 u}{\partial x^2}(n\delta_x, m\delta_\tau) \approx \frac{-u_m^{n-3} + 4u_m^{n-2} - 5u_m^{n-1} + 2u_m^n}{\delta_x^2}.$$

(c) $$\frac{\partial^3 u}{\partial x^3}(n\delta_x, m\delta_\tau) \approx \frac{3u_m^{n-4} - 14u_m^{n-3} + 24u_m^{n-2} - 18u_m^{n-1} + 5u_m^n}{2\delta_x^3}.$$

5. Central difference approximations with error $O(\delta_x^2)$.

(a) $$\frac{\partial u}{\partial x}(n\delta_x, m\delta_\tau) \approx \frac{u_m^{n+1} - u_m^{n-1}}{2\delta_x}.$$

(b) $$\frac{\partial^2 u}{\partial x^2}(n\delta_x, m\delta_\tau) \approx \frac{u_m^{n+1} - 2u_m^n + u_m^{n-1}}{\delta_x^2}.$$

(c) $$\frac{\partial^3 u}{\partial x^3}(n\delta_x, m\delta_\tau) \approx \frac{u_m^{n+2} - 2u_m^{n+1} + 2u_m^{n-1} - u_m^{n-2}}{2\delta_x^3}.$$

6. Central difference approximations with error $O(\delta_x^4)$.

(a) $$\frac{\partial u}{\partial x}(n\delta_x, m\delta_\tau) \approx \frac{-u_m^{n+2} + 8u_m^{n+1} - 8u_m^{n-1} + u_m^{n-2}}{12\delta_x}.$$

(b) $$\frac{\partial^2 u}{\partial x^2}(n\delta_x, m\delta_\tau) \approx \frac{-u_m^{n+2} + 16u_m^{n+1} - 30u_m^n + 16u_m^{n-1} - u_m^{n-2}}{12\delta_x^2}.$$

7. Central difference approximation of a mixed partial derivative

$$\frac{\partial^2 u}{\partial x \partial t}(n\delta_x, m\delta_\tau) = \frac{u_{m+1}^{n+1} - u_{m-1}^{n+1} - u_{m+1}^{n-1} + u_{m-1}^{n-1}}{4\delta_\tau \delta_x} + O\left(\frac{(\delta_x + \delta_\tau)^4}{\delta_\tau \delta_x}\right).$$

The Explicit Finite-difference Method

- Consider the diffusion equation

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2},$$

with boundary conditions

$$u(x, \tau) = u_{-\infty}(x, \tau), \quad u(x, \tau) = u_\infty(x, \tau) \quad \text{as} \quad x \to \pm\infty$$

and initial condition

$$u(x, 0) = u_0(x).$$

- Using a forward-difference estimate for $\partial u/\partial \tau$ (around u_m^n), a central-difference estimate for $\partial^2 u/\partial x^2$ and ignoring $O(\delta_\tau)$ and $O(\delta_x^2)$ terms we obtain (See Fig 8.4, pg 141, WHD)

$$\frac{u_{m+1}^n - u_m^n}{\delta_\tau} = \frac{u_m^{n+1} - 2u_m^n + u_m^{n-1}}{\delta_x^2}. \tag{3}$$

- This may be rearranged as

$$u_{m+1}^n = \kappa u_m^{n-1} + (1 - 2\kappa)u_m^n + \kappa u_m^{n+1}, \tag{4}$$

where κ, known as the Courant number, is given by

$$\kappa = \frac{\delta_\tau}{\delta_x^2}. \tag{5}$$

- If we know the values of u at time step m, we may *explicitly* compute the values at time step $m + 1$; hence the name.

- In simulating the previous equation on a computer we require a finite grid. So, we restrict ourselves to constant x-spacing δ_x over the interval

$$N^- \delta_x \leq x \leq N^+ \delta_x,$$

where $-N^-$ and N^+ are large positive integers. We also restrict ourselves to constant τ-spacing $\delta\tau$ over the interval

$$0 \leq \tau \leq M\delta_\tau,$$

where M is a large positive integer.

- It is then possible to solve the difference equations (4) for $N^- < n < N^+$ and $0 < m \leq M$, using the boundary conditions

$$u_m^{N-} = u_{-\infty}(N^-\delta_x, m\delta_\tau), \quad 0 < m \leq M,$$
$$u_m^{N+} = u_{\infty}(N^+\delta_x, m\delta_\tau), \quad 0 < m \leq M$$

and the initial condition

$$u_0^n = u_0(n\delta_x), \quad N^- \leq n \leq N^+.$$

- Refer to the example BS-calculation on page 143 of handout notes (WHD). Note that, for certain values of κ, the solution may become unstable.
- The stability problem observed in these calculations, results from the fact that there are inevitably round-off errors introduced as a result of finite machine precision. The solution becomes *unstable* if these errors are magnified at each step.

Von Neumann Stability Analysis (Exercise 5)

- Suppose e_m^n are the finite-precision errors introduced into the solution of (4) because of initial errors e_0^n. As a consequence of Fourier analysis, we may assume that the errors take the form

$$e_m^n = \lambda^m \sin(n\omega),$$

for some frequency ω. This means that $e_{m+1}^n = \lambda e_m^n$ and that

$$\begin{aligned} e_m^{n\pm1} &= \lambda^m \sin((n \pm 1)\omega) \\ &= \lambda^m (\sin(n\omega)\cos(\omega) \pm \cos(n\omega)\sin(\omega)) \\ &= \cos(\omega) e_m^n \pm \lambda^m \cos(n\omega)\sin(\omega), \end{aligned}$$

which is derived using the identity $\sin(\alpha \pm \beta) = \sin\alpha\cos\beta \pm \cos\alpha\sin\beta$.

- Substituting these expressions into (4) and simplifying yields

$$\lambda e_m^n = 2\kappa\cos(\omega)e_m^n + (1 - 2\kappa)e_m^n.$$

Solving for λ and using the identity $\cos(\omega) = 1 - 2\sin^2(\omega/2)$ gives

$$\lambda = 1 - 4\kappa\sin^2(\omega/2).$$

- Because the error is expressed in powers of λ, to ensure that the error remains bounded, we require $|\lambda| < 1$ which implies the following restriction on κ

$$0 < \kappa < \frac{1}{2} \leq \frac{1}{2\sin^2(\omega/2)}.$$

The Implicit Finite-difference Method

- Using a backward-difference estimate for $\partial u/\partial \tau$ (around u^n_{m+1}), a central-difference estimate for $\partial^2 u/\partial x^2$ and ignoring terms of $O(\delta_\tau)$ and $O(\delta_x^2)$, we obtain the following expression for the diffusion equation

$$\frac{u^n_{m+1} - u^n_m}{\delta_\tau} = \frac{u^{n+1}_{m+1} - 2u^n_{m+1} + u^{n-1}_{m+1}}{\delta_x^2}. \tag{6}$$

  (See Fig 8.7, pg 145, WHD)

- With κ defined as before (5), this may be rearranged as

$$-\kappa u^{n-1}_{m+1} + (1+2\kappa)u^n_{m+1} - \kappa u^{n+1}_{m+1} = u^n_m.$$

- This defines a system of equations that must be solved in an *implicit* manner to determine the values of u at time $m+1$.

- We may write these equations as the matrix system

$$\begin{bmatrix} 1+2\kappa & -\kappa & 0 & \cdots & 0 \\ -\kappa & 1+2\kappa & -\kappa & \ddots & \vdots \\ 0 & -\kappa & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\kappa \\ 0 & \cdots & 0 & -\kappa & 1+2\kappa \end{bmatrix} \begin{bmatrix} u^{N^-+1}_{m+1} \\ \vdots \\ u^0_{m+1} \\ \vdots \\ u^{N^+-1}_{m+1} \end{bmatrix} = \begin{bmatrix} u^{N^-+1}_m \\ \vdots \\ u^0_m \\ \vdots \\ u^{N^+-1}_m \end{bmatrix} + \kappa \begin{bmatrix} u^{N^-}_{m+1} \\ 0 \\ \vdots \\ 0 \\ u^{N^+}_{m+1} \end{bmatrix}.$$

- If we introduce the $(N^+ - N^- - 1) \times (N^+ - N^- - 1)$ matrix

$$\mathbf{M} = \begin{bmatrix} 1+2\kappa & -\kappa & 0 & \cdots & 0 \\ -\kappa & 1+2\kappa & -\kappa & \ddots & \vdots \\ 0 & -\kappa & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\kappa \\ 0 & \cdots & 0 & -\kappa & 1+2\kappa \end{bmatrix}$$

  and the $(N^+ - N^- - 1)$ length vectors

$$\mathbf{u}_m = \left[u^{N^-+1}_m, \ldots, u^0_m, \ldots, u^{N^+-1}_m\right]^T$$

  and

$$\mathbf{b}_m = \left[u^{N^-}_m, 0, \ldots, 0, u^{N^+}_m\right]^T,$$

  then the system may be rewritten as $\mathbf{M}\mathbf{u}_{m+1} = \mathbf{u}_m + \kappa \mathbf{b}_{m+1}$.
  (NB $\mathbf{b}^m$ is defined differently in WHD.)

- If $\kappa \geq 0$ then $\mathbf{M}$ is invertible, and we have $\mathbf{u}_{m+1} = \mathbf{M}^{-1}(\mathbf{u}_m + \kappa \mathbf{b}_{m+1})$.

- This scheme is Von Neumann stable for $\kappa > 0$. (See Exercise 11, WHD)

The Crank-Nicolson Method

- To overcome the stability limitations of the explicit finite-difference method and to have solutions of order $O(\delta_\tau^2)$ one can create a scheme which is the average of the explicit (3) and implicit (6) schemes.
- Ignoring the $O(\delta_\tau^2)$ and $O(\delta_x^2)$ terms, this leads to the following approximation

$$\frac{u_{m+1}^n - u_m^n}{\delta_\tau} = \frac{1}{2}\left(\frac{u_m^{n+1} - 2u_m^n + u_m^{n-1}}{\delta_x^2} + \frac{u_{m+1}^{n+1} - 2u_{m+1}^n + u_{m+1}^{n-1}}{\delta_x^2}\right)$$

which may be rewritten as

$$u_{m+1}^n - \tfrac{1}{2}\kappa\left(u_{m+1}^{n-1} - 2u_{m+1}^n + u_{m+1}^{n+1}\right) = u_m^n + \tfrac{1}{2}\kappa\left(u_m^{n-1} - 2u_m^n + u_m^{n+1}\right). \quad (7)$$

- Define the $(N^+ - N^- - 1) \times (N^+ - N^- - 1)$ matrices

$$\mathbf{C} = \begin{bmatrix} 1+\kappa & -\frac{1}{2}\kappa & 0 & \cdots & 0 \\ -\frac{1}{2}\kappa & 1+\kappa & -\frac{1}{2}\kappa & \ddots & \vdots \\ 0 & -\frac{1}{2}\kappa & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\frac{1}{2}\kappa \\ 0 & \cdots & 0 & -\frac{1}{2}\kappa & 1+\kappa \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1-\kappa & \frac{1}{2}\kappa & 0 & \cdots & 0 \\ \frac{1}{2}\kappa & 1-\kappa & \frac{1}{2}\kappa & \ddots & \vdots \\ 0 & \frac{1}{2}\kappa & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & \frac{1}{2}\kappa \\ 0 & \cdots & 0 & \frac{1}{2}\kappa & 1-\kappa \end{bmatrix}.$$

- We may then write (7) as $\mathbf{C}\mathbf{u}_{m+1} = \mathbf{D}\mathbf{u}_m + \frac{1}{2}\kappa(\mathbf{b}_m + \mathbf{b}_{m+1})$, with $\mathbf{b}$ as before, which, if $\kappa \geq 0$, gives $\mathbf{u}_{m+1} = \mathbf{C}^{-1}(\mathbf{D}\mathbf{u}_m + \frac{1}{2}\kappa(\mathbf{b}_m + \mathbf{b}_{m+1}))$.
- This scheme is Von Neumann stable for $\kappa > 0$. (See Exercise 17, WHD)

Theta Methods

- The Crank-Nicolson scheme is only one of a family of methods that can be obtained by a suitable weighted average of the Explicit and Implicit finite-difference schemes.
- For $\theta \in [0, 1]$ and κ as in (5), we define the corresponding theta method finite-difference scheme as

$$u_{m+1}^n - \kappa\theta\left(u_{m+1}^{n-1} - 2u_{m+1}^n + u_{m+1}^{n+1}\right) = u_m^n + \kappa(1-\theta)\left(u_m^{n-1} - 2u_m^n + u_m^{n+1}\right).$$

- Note that $\theta = \frac{1}{2}$ corresponds to the Crank-Nicolson scheme (7).
- Using the same mesh parameters as before, define the matrices $\mathbf{C}$ and $\mathbf{D}$ of size $(N^+ - N^- - 1) \times (N^+ - N^- - 1)$ by

$$\mathbf{C} = \begin{bmatrix} 1+2\theta\kappa & -\theta\kappa & 0 & \cdots & 0 \\ -\theta\kappa & 1+2\theta\kappa & -\theta\kappa & \ddots & \vdots \\ 0 & -\theta\kappa & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & -\theta\kappa \\ 0 & \cdots & 0 & -\theta\kappa & 1+2\theta\kappa \end{bmatrix}, \quad \mathbf{D} = \begin{bmatrix} 1-2(1-\theta)\kappa & (1-\theta)\kappa & 0 & \cdots & 0 \\ (1-\theta)\kappa & 1-2(1-\theta)\kappa & (1-\theta)\kappa & \ddots & \vdots \\ 0 & (1-\theta)\kappa & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & (1-\theta)\kappa \\ 0 & \cdots & 0 & (1-\theta)\kappa & 1-2(1-\theta)\kappa \end{bmatrix}.$$

- The system may be rewritten as $\mathbf{C}\mathbf{u}_{m+1} = \mathbf{D}\mathbf{u}_m + \kappa((1-\theta)\mathbf{b}_m + \theta\mathbf{b}_{m+1})$, which gives $\mathbf{u}_{m+1} = \mathbf{C}^{-1}(\mathbf{D}\mathbf{u}_m + \kappa((1-\theta)\mathbf{b}_m + \theta\mathbf{b}_{m+1}))$ with the $\mathbf{b}$ vectors defined as before.

Exercise: Transformation of the Black-Scholes PDE into the Heat Equation

As you know, the Black-Scholes equation and boundary/initial conditions for a call option are

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

with $V(0,t) = 0$, $V(S,t) \approx S$ as $S \to \infty$ and $V(S,T) = \max(S-K, 0)$.

Transform this system into the diffusion equation in two steps, by first making the following change of variables: $S = Ke^x$, $t = T - 2\tau/\sigma^2$ and $V = Kv(x,\tau)$.

Now, transform the resulting system using the change of variables

$$v = e^{\alpha x + \beta \tau} u(x,\tau)$$

with $\alpha = -\frac{1}{2}(\gamma - 1)$ and $\beta = -\frac{1}{4}(\gamma+1)^2$, where $\gamma = 2r/\sigma^2$.

This should result in the diffusion equation

$$\frac{\partial u}{\partial \tau} = \frac{\partial^2 u}{\partial x^2} \qquad \text{for } -\infty < x < \infty, \quad 0 \le \tau \le \sigma^2 T/2,$$

with initial condition

$$u(0,x) = \max(e^{(\gamma+1)x/2} - e^{(\gamma-1)x/2}, 0).$$

Application of the Theta Method to Option Pricing

- Now, we apply the Theta Method to the transformed Black-Scholes PDE to price a call option.
- To make implementation generic for different payoffs and boundary conditions, we define

  $$f_s(x) = Ke^x, \quad f_t(\tau) = T - \frac{2\tau}{\sigma^2}$$

  and

  $$f(V,x,\tau) = \frac{V}{K}\exp(-\alpha x - \beta\tau),$$

  where, as above,

  $$\alpha = -\tfrac{1}{2}(\gamma-1), \quad \beta = -\tfrac{1}{4}(\gamma+1)^2 \quad \text{with} \quad \gamma = 2r/\sigma^2.$$

- Then, the initial condition

  $$u_0(x) = f\left(V_T\left(f_s(x)\right), x, 0\right)$$

  and boundary conditions

  $$u_{-\infty}(x,\tau) = f\left(V^0\left(f_s(x), f_t(\tau)\right), x, \tau\right) \quad \text{and}$$
  $$u_{\infty}(x,\tau) = f\left(V^\infty\left(f_s(x), f_t(\tau)\right), x, \tau\right),$$

  are expressed in terms of the option pay-off and boundary behaviour

  $$V_T(S) = \max(S-K,0), \quad V^0(S,t) = 0 \quad \text{and} \quad V^\infty(S,t) = S - e^{-r(T-t)}K.$$

- We then produce a finite difference estimate of the diffusion solution using the following option related parameters

$$\sigma = 40\%, \quad r = 6\%, \quad K = 50 \quad \text{and} \quad T = 1,$$

with $\theta = 0.5$ (Crank-Nicolson scheme) and mesh parameters

$$N^- = -100, \quad N^+ = 20, \quad M = 35, \quad \delta_x = 0.06 \quad \text{and} \quad \delta_\tau = \frac{\sigma^2 T}{2M}.$$

- After the diffusion solution is computed, prices are obtained by using the transform

$$V_m^n = u_m^n K e^{\alpha n \delta_x + \beta m \delta_\tau}$$

and noting that $V_m^n = V(S_n, t_m)$ where

$$S_n = f_s(n\delta_x) \quad \text{and} \quad t_m = f_t(m\delta_\tau).$$

- The solutions obtained are graphed below and, in a separate plot, the error, being the difference between the Black-Scholes value and the final solution, is graphed.

Example: Black-Scholes Call

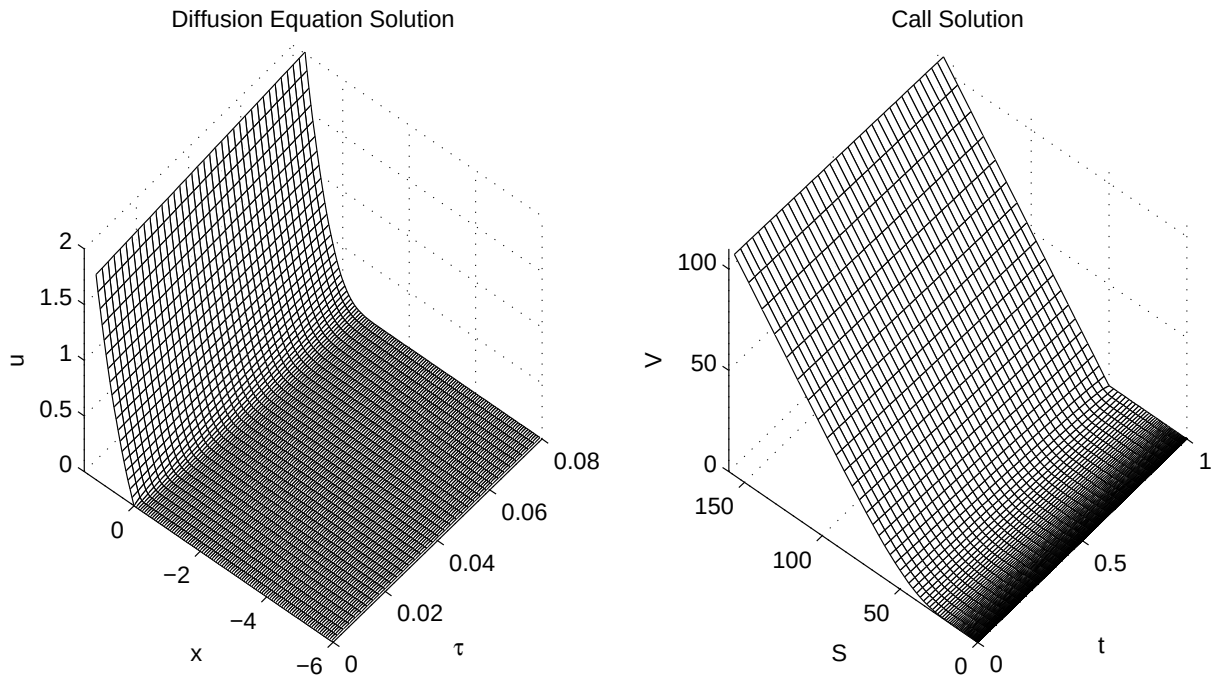


Figure: Diffusion equation solution using transformed call option initial and boundary conditions (left). Finite difference solution for call option (right).

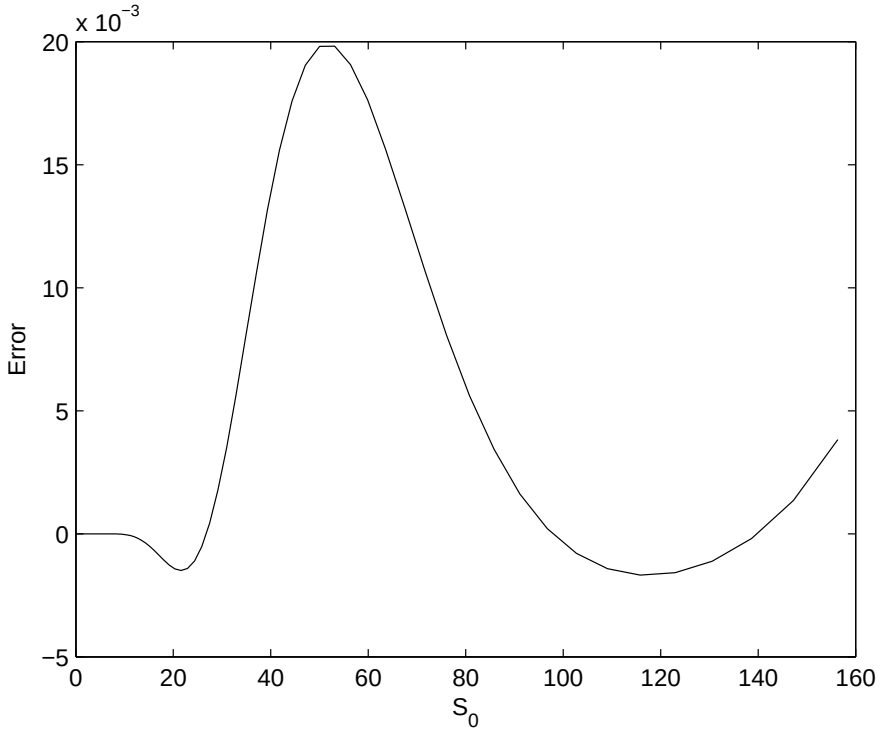


Figure: Difference between Black-Scholes formula and transformed Theta method finite difference solution (at time $t = 0$).

Numerical Methods in Finance II

Lecture 2 - Finite Difference Methods Continued

T.A. McWalter

M.Phil. in Financial Mathematics

2020

Introduction

- One of the problems with using the Black-Scholes PDE transformed into the diffusion equation is that the spacial variable (stock) does not necessarily have conveniently spaced grid points that correspond to the increments or boundaries that may be required for the solution of a particular problem.
- This means that, to obtain prices at specific stock values, we may need to interpolate the transformed diffusion equation solution, thereby introducing another source of error.
- Thus, we now derive and implement a direct theta method finite difference scheme for the Black-Scholes PDE, which allows the specification of a grid in terms of stock-prices.
- This is achieved by working with a time-reversed Black-Sholes PDE and initially specifying both the Implicit and Explicit formulations from which the theta method may be derived.
- We then price a double-barrier call option using the approach.
- Finally, we investigate efficient computational methods for solutions of (tridiagonal) linear systems of equations by introducing the Thomas algorithm, the Jacobi and Gauss-Seidel methods, and successive over relaxation (SOR) methods.

- Consider, once again, the Black-Scholes equation

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

  where, for now, we leave the initial and boundary conditions unspecified.

- We now specify a finite-difference scheme that directly represents the Black-Scholes PDE.

- For convenience, the only transformation we make is a reversal of time, $\tau = T - t$, in order to change the terminal condition into an initial condition.

- The time-reversed Black-Scholes equation is then

$$\frac{\partial U}{\partial \tau} - rS\frac{\partial U}{\partial S} - \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 U}{\partial S^2} + rU = 0,$$

  with suitable initial and boundary conditions.

- In principle, we need to work on an infinite domain since $S \in [0, \infty)$. However, we truncate the domain to $[S_{\min}, S_{\max}]$, for $0 \leq S_{\min} < S_{\max}$.

- We use a regular mesh

$$\{(S_{\min} + n\delta_s, m\delta_\tau) : 0 \leq n \leq N, 0 \leq m \leq M\},$$

  for suitably large values $M, N \in \mathbb{N}$, with $\delta_s = (S_{\max} - S_{\min})/N$ and $\delta_\tau = T/M$.

- Using a forward and backward difference for the time derivatives and central differences for the spatial derivatives we obtain the following discretisations for the explicit scheme

$$\frac{U_{m+1}^n - U_m^n}{\delta_\tau} - r(S_{\min} + n\delta_s)\frac{U_m^{n+1} - U_m^{n-1}}{2\delta_s} - \frac{1}{2}\sigma^2 (S_{\min} + n\delta_s)^2 \frac{U_m^{n+1} - 2U_m^n + U_m^{n-1}}{\delta_s^2} + rU_m^n = 0$$

  and the implicit scheme

$$\frac{U_{m+1}^n - U_m^n}{\delta_\tau} - r(S_{\min} + n\delta_s)\frac{U_{m+1}^{n+1} - U_{m+1}^{n-1}}{2\delta_s} - \frac{1}{2}\sigma^2 (S_{\min} + n\delta_s)^2 \frac{U_{m+1}^{n+1} - 2U_{m+1}^n + U_{m+1}^{n-1}}{\delta_s^2} + rU_{m+1}^n = 0$$

- **Show as an exercise** that these equations may be written in matrix form as

$$\mathbf{U}_{m+1} = \mathbf{F}\mathbf{U}_m + \mathbf{b}_m \tag{1}$$
$$\mathbf{G}\mathbf{U}_{m+1} = \mathbf{U}_m + \mathbf{b}_{m+1}, \tag{2}$$

  for $0 \leq m < M-1$, where $\mathbf{U}_m, \mathbf{U}_{m+1} \in \mathbb{R}^{(N-1)}$ are the solutions at times m and $m+1$, $\mathbf{b}_m \in \mathbb{R}^{(N-1)}$ is a vector specifying boundary conditions and the matrices $\mathbf{F}$ and $\mathbf{G}$ are $(N-1) \times (N-1)$ tridiagonal.

- In the above, the tridiagonal matrices

$$\mathbf{F} = (1 - r\delta_\tau)\mathbf{I} + \frac{1}{2}r\delta_\tau \mathbf{D}_1\mathbf{T}_1 + \frac{1}{2}\sigma^2\delta_\tau \mathbf{D}_2\mathbf{T}_2 \quad \text{and} \quad \mathbf{G} = 2\mathbf{I} - \mathbf{F}$$

  are given in terms of the identity matrix $\mathbf{I}$, the diagonal matrices

$$\mathbf{D}_1 = \mathrm{diag}(S_{\min}/\delta_s + [1, 2, \ldots, N-1]), \quad \mathbf{D}_2 = \mathbf{D}_1^2$$

  and the $(N-1) \times (N-1)$ tridiagonal matrices

$$\mathbf{T}_1 = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \\ -1 & 0 & 1 & \ddots & \vdots \\ 0 & -1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & -1 & 0 \end{bmatrix}, \quad \mathbf{T}_2 = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix}.$$

- The $(N-1) \times 1$ boundary condition vectors are given by

$$\mathbf{b}_m = \begin{bmatrix} \frac{1}{2}\delta_\tau(S_{\min}/\delta_s + 1)(\sigma^2(S_{\min}/\delta_s + 1) - r)U_m^0 \\ 0 \\ \vdots \\ 0 \\ \frac{1}{2}\delta_\tau(S_{\max}/\delta_s - 1)(\sigma^2(S_{\max}/\delta_s - 1) + r)U_m^N \end{bmatrix},$$

  with $U_m^0 = U^0(S_{\min}, m\delta_\tau)$ and $U_m^N = U^\infty(S_{\max}, m\delta_\tau)$.

- Of course, to solve this system, one must also specify the initial condition

$$U_0^n = U_0(S_{\min} + n\delta_s).$$

- Using the above, we are now in a position to derive the Theta method.
- Multiplying (2) by θ and (1) by $(1-\theta)$ and adding the two gives

$$(\theta\mathbf{G} + (1-\theta)\mathbf{I})\mathbf{U}_{m+1} = ((1-\theta)\mathbf{F} + \theta\mathbf{I})\mathbf{U}_m + (1-\theta)\mathbf{b}_m + \theta\mathbf{b}_{m+1}.$$

  Solving for $\mathbf{U}_{m+1}$ gives

$$\mathbf{U}_{m+1} = (\theta\mathbf{G} + (1-\theta)\mathbf{I})^{-1}[((1-\theta)\mathbf{F} + \theta\mathbf{I})\mathbf{U}_m + (1-\theta)\mathbf{b}_m + \theta\mathbf{b}_{m+1}].$$

- Although we do not show it here, this scheme can be shown to be unconditionally stable for $\frac{1}{2} \leq \theta \leq 1$, where $\theta = 1$ corresponds to the fully implicit scheme and $\theta = \frac{1}{2}$ corresponds to the Crank-Nicolson scheme. The fully explicit scheme corresponds to $\theta = 0$, but is not recommended since it is generally unstable.

- **Example:** We now price a double barrier up-and-out/down-and-out call option with strike K and lower and upper boundaries l and u. Firstly, note that, for $l < S < u$, the initial condition is given by

$$U_0(S) = V_T(S) = \max(S - K, 0),$$

while the boundary conditions, which implement the knock-out features, are given by

$$U^0(l, \tau) = V^0(l, T - \tau) = 0$$

and

$$U^\infty(u, \tau) = V^\infty(u, T - \tau) = 0.$$

- The valuation surface is computed using the following option related parameters

$$\sigma = 40\%, \quad r = 6\%, \quad K = 40, \quad l = 20, \quad u = 100, \quad T = 0.75$$

and mesh related parameters

$$S_{\min} = l, \quad S_{\max} = u, \quad N = M = 40.$$

Example: Double Barrier Call Option

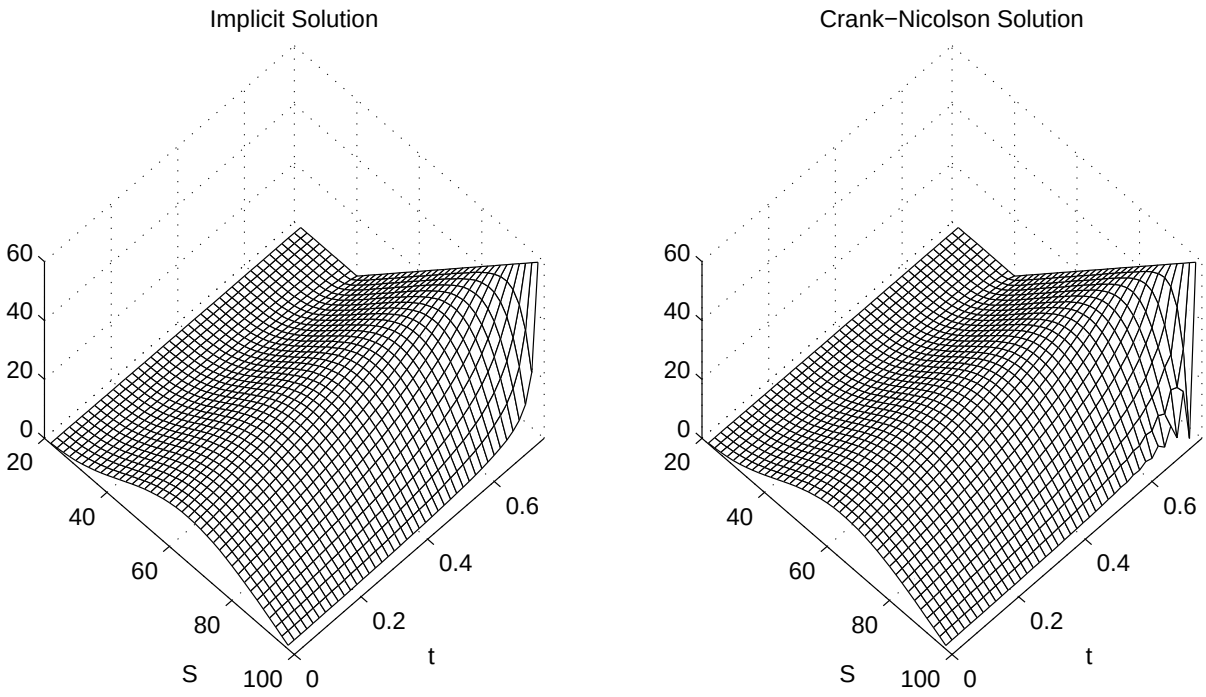


Figure: Output from the Theta finite-difference method on the Black-Scholes PDE. The left graph corresponds to $\theta = 1$ while the right corresponds to $\theta = 0.5$.

- The semi-analytical price for this double barrier call option on $l < S < u$ is given by

$$C = \sum_{n=-\infty}^{+\infty} S\left[\left(\frac{u^n}{l^n}\right)^{\mu}[\Phi(d_1) - \Phi(d_2)] - \left(\frac{l^{n+1}}{u^n S}\right)^{\mu}[\Phi(d_3) - \Phi(d_4)]\right] -$$
$$Ke^{-rT}\left[\left(\frac{u^n}{l^n}\right)^{\mu-2}[\Phi(d_1 - \sigma\sqrt{T}) - \Phi(d_2 - \sigma\sqrt{T})] -\right.$$
$$\left.\left(\frac{l^{n+1}}{u^n S}\right)^{\mu-2}[\Phi(d_3 - \sigma\sqrt{T}) - \Phi(d_4 - \sigma\sqrt{T})]\right]$$

where

$$\mu = \frac{2r}{\sigma^2} + 1$$
$$d_1 = \frac{\log(Su^{2n}/(Kl^{2n})) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_2 = \frac{\log(Su^{2n-1}/l^{2n}) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_3 = \frac{\log(l^{2n+2}/(KSu^{2n})) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}$$
$$d_4 = \frac{\log(l^{2n+2}/(Su^{2n+1})) + (r + \sigma^2/2)T}{\sigma\sqrt{T}}.$$

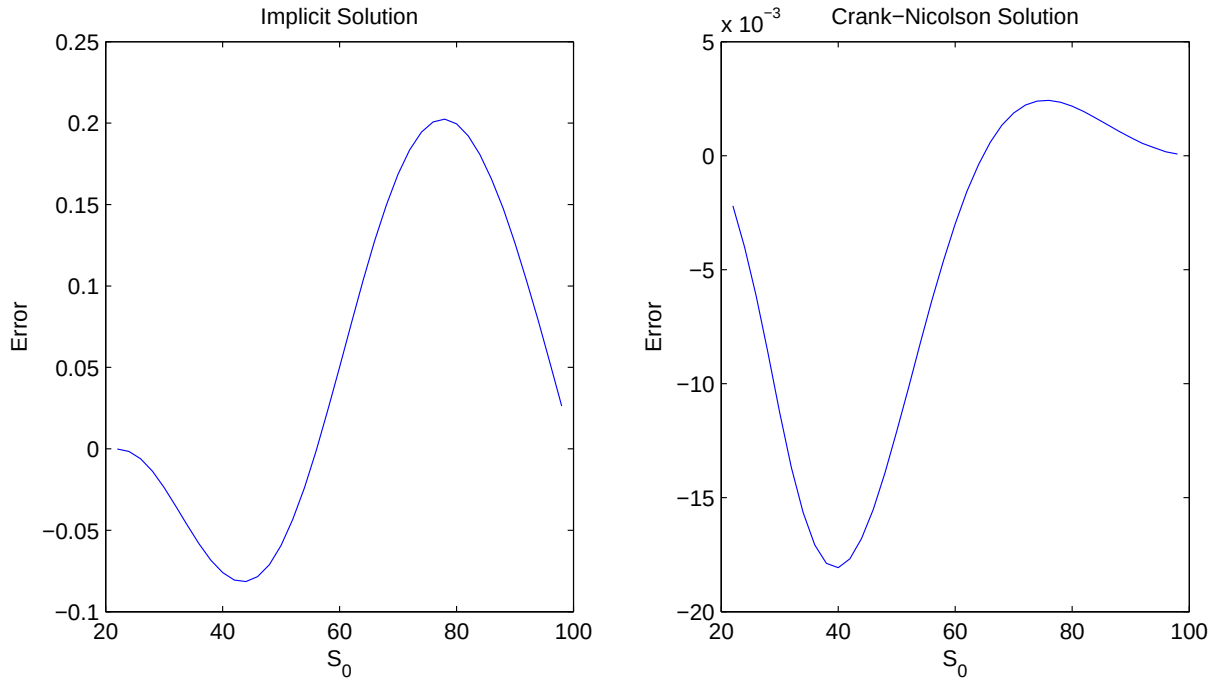


Figure: Difference between the semi-anaytical solution (truncated on $n \in [-100, 100]$) and the Theta finite-difference solution for the option price at time $t = 0$.

Linear Algebra: LU Decomposition

- With the exception of the explicit methods, the finite-difference methods we have looked at require the solution of large systems of sparse linear algebraic equations. In particular, all the systems have been tridiagonal.
- When coding in languages other than Matlab, it is necessary to provide efficient algorithms to do the matrix inversions.
- Here we explore the use of the Thomas Algorithm for solutions of such systems.

Theorem 1.2 (Thomas algorithm) *Given a tridiagonal non-singular $n \times n$ matrix $\mathbf{A}$ we may compute its LU decomposition $\mathbf{A} = \mathbf{LU}$ as*

$$\begin{bmatrix} a_1 & c_1 & 0 & \cdots & 0 \\ b_1 & a_2 & c_2 & \ddots & \vdots \\ 0 & b_2 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & c_{n-1} \\ 0 & \cdots & 0 & b_{n-1} & a_n \end{bmatrix} = \begin{bmatrix} 1 & 0 & \cdots & \cdots & 0 \\ l_1 & 1 & \ddots & & \vdots \\ 0 & l_2 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & l_{n-1} & 1 \end{bmatrix} \begin{bmatrix} d_1 & u_1 & 0 & \cdots & 0 \\ 0 & d_2 & u_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ \vdots & & \ddots & \ddots & u_{n-1} \\ 0 & \cdots & \cdots & 0 & d_n \end{bmatrix},$$

where, for $1 \leq i \leq n-1$, the matrix elements may be computed by

$$d_1 = a_1, \qquad d_{i+1} = a_{i+1} - \frac{c_i b_i}{d_i}, \qquad u_i = c_i, \qquad l_i = \frac{b_i}{d_i}.$$

- Prove this as an exercise.

- Given that the $n \times n$ matrix $\mathbf{A} = \mathbf{LU}$ is decomposed using the above theorem, it is now possible to provide a very efficient algorithm to solve the system $\mathbf{Ax} = \mathbf{y}$ for $\mathbf{x}$.
- The original problem may be rewritten as $\mathbf{L}(\mathbf{Ux}) = \mathbf{y}$ and subsequently split into two problems that must be solved in order; firstly to solve the intermediate problem $\mathbf{Lz} = \mathbf{y}$ for $\mathbf{z}$, and then to solve $\mathbf{Ux} = \mathbf{z}$ for $\mathbf{x}$.
- The intermediate solution $\mathbf{z}$ may be computed using *forward substitution* as follows:

$$z_1 = y_1 \qquad \text{and} \qquad z_i = y_i - l_{i-1} z_{i-1} \qquad \text{for} \qquad 2 \leq i \leq n.$$

- Finally, the solution can be computed using *backward substitution* as follows:

$$x_n = \frac{z_n}{d_n} \qquad \text{and} \qquad x_i = \frac{z_i - u_i x_{i+1}}{d_i} \qquad \text{for} \qquad n-1 \geq i \geq 1.$$

- This algorithm solves the system in $O(n)$ steps as opposed to the usual $O(n^3)$ steps for Gaussian elimination.

Linear algebra: SOR

- We wish to solve the general $n \times n$ system $\mathbf{Ax} = \mathbf{y}$

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ a_{n,1} & \cdots & a_{n,n-1} & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{bmatrix}.$$

- Initially, we may rewrite this system as

$$\begin{bmatrix} a_{1,1} & 0 & \cdots & 0 \\ 0 & a_{2,2} & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \cdots & 0 & a_{n,n} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} y_1 \\ \vdots \\ \vdots \\ y_n \end{bmatrix} - \begin{bmatrix} 0 & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & a_{n-1,n} \\ a_{n,1} & \cdots & a_{n,n-1} & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ \vdots \\ x_n \end{bmatrix}.$$

- The diagonal is easily invertible, which suggests the following iterative scheme

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - \sum_{j=1, j \neq i}^{n} a_{i,j} x_j^{(k)} \right), \tag{3}$$

  for $1 \leq i \leq n$, where the superscript on x indicates the iteration index.
- This requires an initial guess $\mathbf{x}^{(0)}$, with convergence reached when $|x_i^{(k+1)} - x_i^{(k)}| < \mathrm{tol}$ for all i.
- This iterative approach is known as the *Jacobi method*.

- If we write $\mathbf{A} = \mathbf{D} + \mathbf{U} + \mathbf{L}$ where $\mathbf{D}$ is the diagonal of $\mathbf{A}$, and $\mathbf{U}$ and $\mathbf{L}$ are the upper and lower triangular matrices of $\mathbf{A}$ (note, this is not the $\mathbf{LU}$ decomposition), then we can rewrite the above in a compact matrix form as

$$\mathbf{x}^{(k+1)} = \mathbf{D}^{-1}\left(\mathbf{y} - (\mathbf{U} + \mathbf{L})\mathbf{x}^{(k)}\right).$$

- To improve convergence, it is possible to use the new updates of $x_i^{(k+1)}$ as soon as they are computed. Then (3) may be written as follows

$$x_i^{(k+1)} = \frac{1}{a_{i,i}} \left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{i,j} x_j^{(k)} \right), \tag{4}$$

  for $1 \leq i \leq n$.
- This is known as the *Gauss-Seidel method*.
- Using the matrix notation above, it may be written as

$$\mathbf{x}^{(k+1)} = (\mathbf{D} + \mathbf{L})^{-1}\left(\mathbf{y} - \mathbf{U}\mathbf{x}^{(k)}\right).$$

- Because, in our case, the matrix $\mathbf{A}$ comes from a finite-difference scheme for a parabolic PDE, it can be shown that this solution converges monotonically. This means that the sign of $\Delta_i^{(k)} = x_i^{(k+1)} - x_i^{(k)}$ stays the same as k increases.
- This can be exploited by the method of successive over-relaxation or SOR, which weights this difference by a factor larger than one to accelerate convergence.

- To see this, by adding back the $x_i^{(k)}$ term on the left-hand side, write (4) as

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

- We can then weight the factor $\Delta_i^{(k)}$, which corresponds to the term on the left of this equation, by an acceleration or over-relaxation parameter ω. This gives

$$x_i^{(k+1)} = x_i^{(k)} + \frac{\omega}{a_{i,i}} \left(y_i - \sum_{j=1}^{i-1} a_{i,j} x_j^{(k+1)} - \sum_{j=i}^{n} a_{i,j} x_j^{(k)} \right).$$

- In matrix form, this may be written as

$$\mathbf{x}^{(k+1)} = (\mathbf{D} + \omega \mathbf{L})^{-1} \big(\omega \mathbf{y} + ((1-\omega)\mathbf{D} - \omega \mathbf{U}) \mathbf{x}^{(k)} \big). \tag{5}$$

- It can be shown, provided $0 < \omega < 2$, that the above formulation converges for the finite difference systems we have considered. Values of $0 < \omega < 1$ are applicable for oscillatory systems, in which case ω is known as an under-relaxation parameter. When $1 < \omega < 2$ it is known as an over-relaxation parameter — this is the case relevant to us.
- It is difficult, and problem specific, to derive optimal values for ω. A simple approach is to start with $\omega = 1$ and to perform the SOR routine adding small positive increments to ω (say, 0.05) until the number of iterations required for convergence no longer decreases.

Numerical Methods in Finance II

Lecture $*$ (optional) - Finite Difference Methods for Two Underlyings

T.A. McWalter

M.Phil. in Financial Mathematics

2020

Introduction

- Having looked at finite-difference schemes for pricing claims based on one asset, we now turn our attention to pricing claims expressed as functions of two underlying assets.
- The approach taken is to work in a transformed solution based on time-reversed Brownian motions.
- To keep things manageable, we only consider the case where the underlying Brownian motions are uncorrelated. There are, however, simple extensions that allow for correlation, albeit at the expense of complicated boundary solution specifications.
- Initially, we shall formulate an explicit method solution to the problem. This approach is conditionally stable.
- Later, to achieve unconditional stability, we investigate the use of an alternate direction implicit (ADI) formulation of the problem.
- As an example, we shall consider the problem of pricing a European basket put option (for which there is no closed-form solution).

Model Specification

- Consider the time t price of a European option, given in terms of two geometric Brownian motion stock prices, S^1 and S^2,

$$h(S_t^1, S_t^2, t) = e^{-r(T-t)}\mathbb{E}[H(S_T^1, S_T^2, T)|\mathscr{F}_t],$$

where H is the payoff function. Note that the expectation is taken under the risk-neutral measure.

- We may rewrite this in terms of the underlying (uncorrelated) Brownian motions, W_t^1 and W_t^2, and $\tau = T - t$

$$u(W_{T-\tau}^1, W_{T-\tau}^2, \tau) = e^{-r\tau}\mathbb{E}[\varphi(W_T^1, W_T^2, T)|\mathscr{F}_{T-\tau}],$$

where

$$\varphi(x_1, x_2, T) = H\left(S_0^1 e^{(r-\sigma_1^2/2)T+\sigma_1 x_1}, S_0^2 e^{(r-\sigma_2^2/2)T+\sigma_2 x_2}, T\right).$$

- The function h is governed by a bivariate Black-Scholes equation specified in terms of the stocks, while the function u solves the following bivariate (time reversed) heat equation

$$\frac{\partial u(x_1, x_2, \tau)}{\partial \tau} - \frac{1}{2}\frac{\partial^2 u(x_1, x_2, \tau)}{\partial x_1^2} - \frac{1}{2}\frac{\partial^2 u(x_1, x_2, \tau)}{\partial x_2^2} + ru(x_1, x_2, \tau) = 0.$$

- The model above assumes that the stock prices are uncorrelated. It is possible to incorporate correlation, ρ, by considering φ of the form

$$\varphi(x_1, x_2, T) = H\left(S_0^1 e^{(r-\sigma_1^2/2)T+\sigma_1 x_1}, S_0^2 e^{(r-\sigma_2^2/2)T+\sigma_2\left(\rho x_1+\sqrt{1-\rho^2}x_2\right)}, T\right).$$

We shall, however, avoid this formulation for now as it complicates the specification of boundary conditions (in terms of x_1 and x_2). See Crépey (2013) for (some) further details.

- We then consider a discretised version of the function

$$u_m^{i,j} = u(i\delta_1, j\delta_2, m\delta_\tau)$$

for Brownian motion increments δ_1 and δ_2, and time increment δ_τ, with

$$N_1^- \leq i \leq N_1^+, \quad N_2^- \leq j \leq N_2^+ \quad \text{and} \quad 0 \leq m \leq M.$$

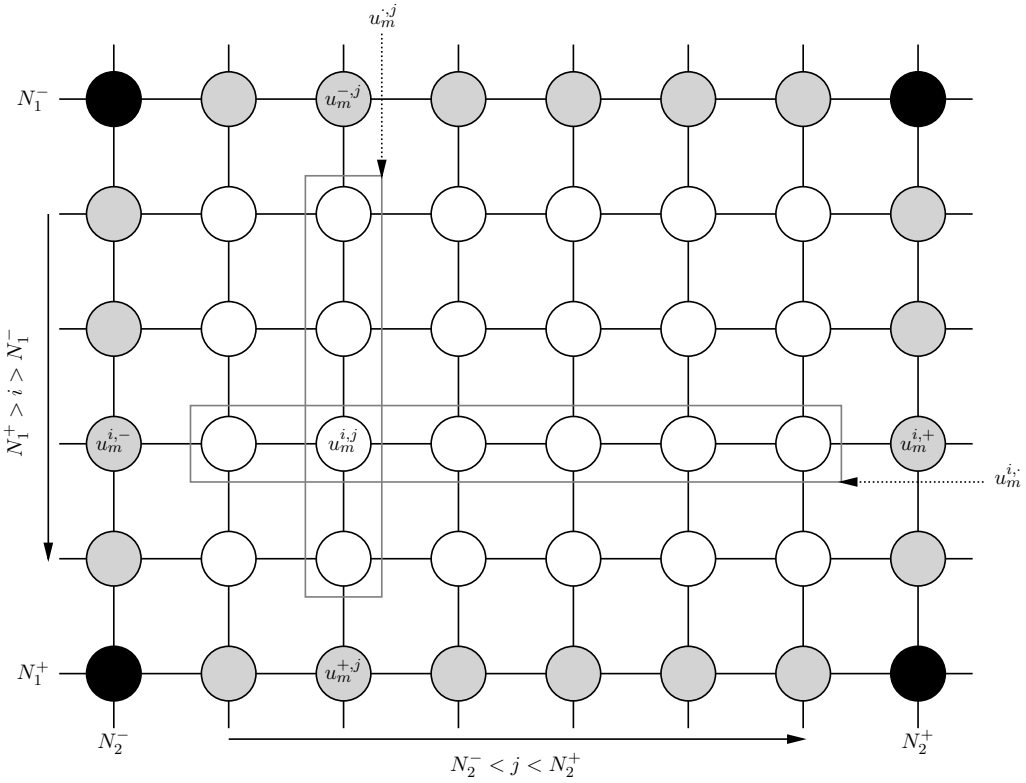
- Aside from the initial condition, specified by the payoff

$$u_0^{i,j} = \varphi(i\delta_1, j\delta_2, T),$$

we also require four boundary conditions, of the form $\phi(x_1, x_2, t)$, specified as the price, at time t, in terms of $W_t^1 = x_1$ and $W_t^2 = x_2$:

$$\begin{aligned}
u_m^{-,j} &= \phi_{S^1}^-(N_1^-\delta_1, j\delta_2, T - m\delta_\tau) && S_t^1 \approx 0\\
u_m^{+,j} &= \phi_{S^1}^+(N_1^+\delta_1, j\delta_2, T - m\delta_\tau) && S_t^1 \text{ is large}\\
u_m^{i,-} &= \phi_{S^2}^-(i\delta_1, N_2^-\delta_2, T - m\delta_\tau) && S_t^2 \approx 0\\
u_m^{i,+} &= \phi_{S^2}^+(i\delta_1, N_2^+\delta_2, T - m\delta_\tau) && S_t^2 \text{ is large.}
\end{aligned}$$

- The following diagram shows a schematic indicating the matrix layout of the solution at time slice m.
- The solution consists of an $(N_1^+ - N_1^- - 1) \times (N_2^+ - N_2^- - 1)$ matrix at each time step m — this is represented by the white cells. The grey cells indicate boundary conditions at time step m.



Explicit Finite-difference Scheme

- Using a forward difference for the time derivative and central differences for the spacial derivatives (around $u_m^{i,j}$) we have

$$\frac{u_{m+1}^{i,j} - u_m^{i,j}}{\delta_\tau} - \frac{1}{2}\frac{u_m^{i+1,j} - 2u_m^{i,j} + u_m^{i-1,j}}{\delta_1^2} - \frac{1}{2}\frac{u_m^{i,j+1} - 2u_m^{i,j} + u_m^{i,j-1}}{\delta_2^2} + ru_m^{i,j} = 0.$$

- Solving for $u_{m+1}^{i,j}$ gives the following system

$$u_{m+1}^{i,j} = \tfrac{1}{2}\kappa_1(u_m^{i+1,j} - 2u_m^{i,j} + u_m^{i-1,j}) + \tfrac{1}{2}\kappa_2(u_m^{i,j+1} - 2u_m^{i,j} + u_m^{i,j-1}) + (1 - r\delta_\tau)u_m^{i,j},$$

where $\kappa_1 = \delta_\tau/\delta_1^2$ and $\kappa_2 = \delta_\tau/\delta_2^2$.

- Although we do not attempt it here, this system can be shown to be Von Neumann stable if $\frac{1}{2}\kappa_1 + \frac{1}{2}\kappa_2 < \frac{1}{2}$. (See Duffy (2006)).
- We can express the system in vector/matrix form as

$$u_{m+1}^{\cdot,j} = \tfrac{1}{2}\kappa_1(\mathbf{F}_1 u_m^{\cdot,j} + \mathbf{b}_m^j) + \tfrac{1}{2}\kappa_2(u_m^{\cdot,j+1} - 2u_m^{\cdot,j} + u_m^{\cdot,j-1}) + (1 - r\delta_\tau)u_m^{\cdot,j},$$

for $N_2^- < j < N_2^+$. (See below for definition of $\mathbf{F}_1$ and $\mathbf{b}_m^j$).

- Note that when $j = N_2^- + 1$ then $u_m^{\cdot,j-1} = u_m^{\cdot,-}$ and when $j = N_2^+ - 1$ then $u_m^{\cdot,j+1} = u_m^{\cdot,+}$. These are the boundary conditions in vector form.

- In the above,

$$\mathbf{F}_1 = \begin{bmatrix} -2 & 1 & 0 & \cdots & 0 \\ 1 & -2 & 1 & \ddots & \vdots \\ 0 & 1 & \ddots & \ddots & 0 \\ \vdots & \ddots & \ddots & \ddots & 1 \\ 0 & \cdots & 0 & 1 & -2 \end{bmatrix} \quad \text{and} \quad \mathbf{b}_m^j = \begin{bmatrix} u_m^{-,j} \\ 0 \\ \vdots \\ 0 \\ u_m^{+,j} \end{bmatrix},$$

  are a tridiagonal matrix of size $(N_1^+ - N_1^- - 1) \times (N_1^+ - N_1^- - 1)$ and a vector of size $(N_1^+ - N_1^- - 1) \times 1$.

- In this formulation, we have chosen to specify the vector/matrix formulation in terms of the columns of the solution — we could just as well have specified it in terms of the rows of the solution.

- In this case we would have

$$u_{m+1}^{i,\cdot} = \tfrac{1}{2}\kappa_1(u_m^{i+1,\cdot} - 2u_m^{i,\cdot} + u_m^{i-1,\cdot}) + \tfrac{1}{2}\kappa_2(u_m^{i,\cdot}\mathbf{F}_2 + \mathbf{b}_m^i) + (1 - r\delta_\tau)u_m^{i,\cdot},$$

  for $N_1^- < i < N_1^+$, where $\mathbf{F}_2$ is a tridiagonal matrix, with the same elements as $\mathbf{F}_1$, of size $(N_2^+ - N_2^- - 1) \times (N_2^+ - N_2^- - 1)$ and $\mathbf{b}_m^i = [u_m^{i,-}, 0, \cdots, 0, u_m^{i,+}]$ is of size $1 \times (N_2^+ - N_2^- - 1)$.

- As in the previous case, vector boundary conditions apply. When $i = N_1^- + 1$ then $u_m^{i-1,\cdot} = u_m^{-,\cdot}$, and when $i = N_1^+ - 1$ then $u_m^{i+1,\cdot} = u_m^{+,\cdot}$.

Example: Basket option

- Consider a European basket option in terms of stocks S^1 and S^2 and strike K with payoff $H(S_T^1, S_T^2, T) = \max(K - S_T^1 - S_T^2, 0)$.
- The payoff (initial condition) is then

$$\varphi(x_1, x_2, T) = \max(K - f_1(x_1, T) - f_2(x_2, T), 0),$$

  where

$$f_i(x, t) = S_0^i e^{(r - \sigma_i^2/2)t + \sigma_i x}.$$

- The boundary conditions are

$$\begin{aligned} \phi_{S^1}^-(x_1, x_2, t) &= \mathsf{BS}_{\text{put}}(f_2(x_2, t), t, \sigma_2, r, K, T) \\ \phi_{S^1}^+(x_1, x_2, t) &= 0 \\ \phi_{S^2}^-(x_1, x_2, t) &= \mathsf{BS}_{\text{put}}(f_1(x_1, t), t, \sigma_1, r, K, T) \\ \phi_{S^2}^+(x_1, x_2, t) &= 0, \end{aligned}$$

- The graph below shows the price at inception of the basket option as a function of stock prices for the option related parameters

$$S_0^1 = 6, \quad S_0^2 = 4, \quad \sigma_1 = 30\%, \quad \sigma_2 = 40\%, \quad K = 8, \quad T = 1 \quad \text{and} \quad r = 6\%,$$

  and the mesh related parameters

$$M = 100, \quad N_1^- = -140, \quad N_1^+ = 30, \quad N_2^- = -100, \quad N_2^+ = 25,$$
$$\delta_1 = \delta_2 = 0.2 \quad \text{and} \quad \delta_\tau = T/M.$$

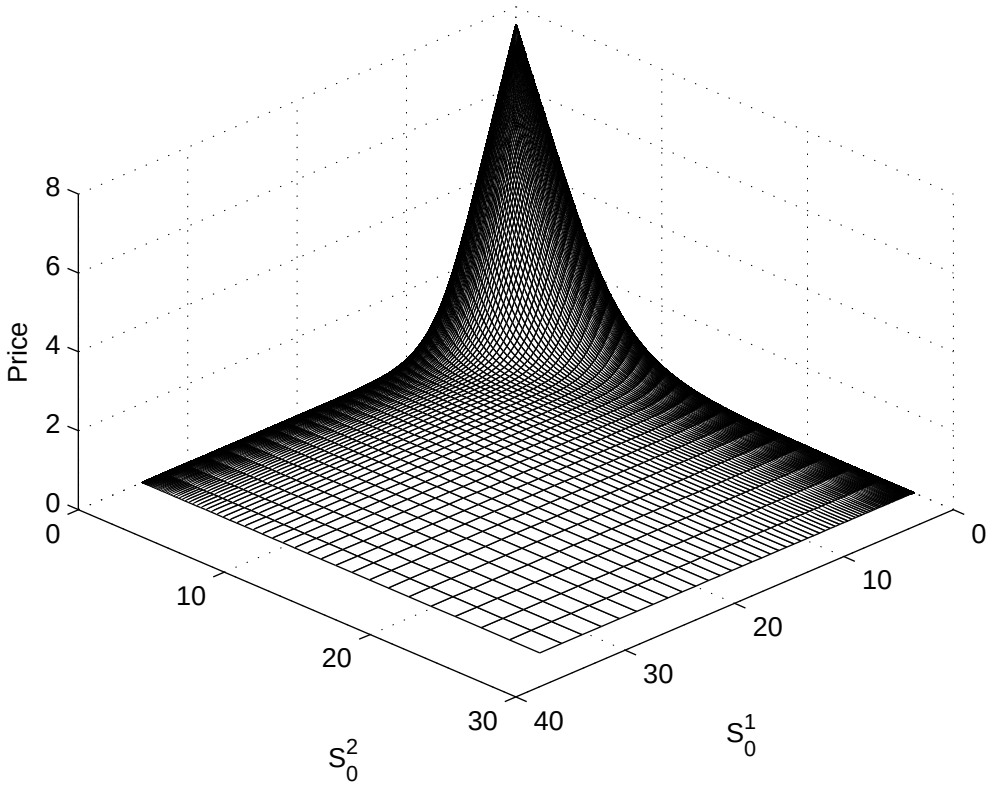


Figure: Price of a European basket put option at inception.

The Alternating Direction Implicit method

- We have seen that the Explicit method of the previous section is only conditionally stable. We now introduce an approach that is both computationally tractable and unconditionally stable. For a proof of stability, see Duffy (2006).
- The idea is to introduce an extra time slice $(m+1/2)$ between times m and $m+1$.
- Between time m and $m+1/2$, solve the system, implicit in the x_1 direction and explicit in the x_2, direction using

$$\frac{u_{m+1/2}^{i,j}-u_m^{i,j}}{\delta_\tau/2}-\frac{1}{2}\frac{u_{m+1/2}^{i+1,j}-2u_{m+1/2}^{i,j}+u_{m+1/2}^{i-1,j}}{\delta_1^2} - \frac{1}{2}\frac{u_m^{i,j+1}-2u_m^{i,j}+u_m^{i,j-1}}{\delta_2^2}+\frac{r}{2}(u_m^{i,j}+u_{m+1/2}^{i,j})=0.$$

- Then, between time $m+1/2$ and $m+1$, solve the system, explicit in the x_1 direction and implicit in the x_2, direction using

$$\frac{u_{m+1}^{i,j}-u_{m+1/2}^{i,j}}{\delta_\tau/2}-\frac{1}{2}\frac{u_{m+1/2}^{i+1,j}-2u_{m+1/2}^{i,j}+u_{m+1/2}^{i-1,j}}{\delta_1^2} - \frac{1}{2}\frac{u_{m+1}^{i,j+1}-2u_{m+1}^{i,j}+u_{m+1}^{i,j-1}}{\delta_2^2}+\frac{r}{2}(u_{m+1/2}^{i,j}+u_{m+1}^{i,j})=0.$$

- **Exercise:** Show that the above two systems may be written in vector/matrix form as

$$u^{\cdot,j}_{m+1/2} = \mathbf{G}_1^{-1}\left[\frac{\kappa_2}{4}(u^{\cdot,j+1}_m - 2u^{\cdot,j}_m + u^{\cdot,j-1}_m) + \left(1 - \frac{r\delta_\tau}{4}\right)u^{\cdot,j}_m + \frac{\kappa_1}{4}\mathbf{b}^j_{m+1/2}\right]$$

for $N_2^- < j < N_2^+$ and

$$u^{i,\cdot}_{m+1} = \left[\frac{\kappa_1}{4}(u^{i+1,\cdot}_{m+1/2} - 2u^{i,\cdot}_{m+1/2} + u^{i-1,\cdot}_{m+1/2}) + \left(1 - \frac{r\delta_\tau}{4}\right)u^{i,\cdot}_{m+1/2} + \frac{\kappa_2}{4}\mathbf{b}^i_{m+1}\right]\mathbf{G}_2^{-1}$$

for $N_1^- < i < N_1^+$, where $\mathbf{b}^i_m$ and $\mathbf{b}^j_m$ are defined as before,

$$\mathbf{G}_1 = \left(1 + \frac{r\delta_\tau}{4}\right)\mathbf{I} - \frac{\kappa_1}{4}\mathbf{F}_1 \quad \text{and} \quad \mathbf{G}_2 = \left(1 + \frac{r\delta_\tau}{4}\right)\mathbf{I} - \frac{\kappa_2}{4}\mathbf{F}_2,$$

where $\mathbf{I}$, in each case, is an identity matrix of the correct size.

- As before, the boundary vectors are specified as

$$\begin{aligned}
u^{\cdot,j-1}_m &= u^{\cdot,-}_m && \text{when } j = N_2^- + 1,\\
u^{\cdot,j+1}_m &= u^{\cdot,+}_m && \text{when } j = N_2^+ - 1,\\
u^{i-1,\cdot}_{m+1/2} &= u^{-,\cdot}_{m+1/2} && \text{when } i = N_1^- + 1 \quad \text{and}\\
u^{i+1,\cdot}_{m+1/2} &= u^{+,\cdot}_{m+1/2} && \text{when } i = N_1^+ - 1.
\end{aligned}$$

- Below, we show the price of the European basket option with the same parameters as the previous example, using $M = 20$. Note that even though the method is more complex, it is quicker due to the reduction in M.

Example: ADI Solution of the Basket option