Discretization Schemes for PDEs

Goals:

- Discuss common discretization schemes for solving PDEs numerically
- Discuss the stability of these discretization schemes

Relevant literature:

- Hirsa, Chapters 3 & 4

Discretizing the Black-Scholes PDE

As you remember from the last lecture, one way to discretize the Black-Scholes PDE

$$\frac{\partial c}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 c}{\partial s^2} + rs\frac{\partial c}{\partial s} - rc = 0 \tag{1}$$

is to use backward differences in t:

$$\begin{aligned} & \frac{C(s_i, t_j) - C(s_i, t_{j-1})}{h_t} \\ + \quad & \frac{1}{2}\sigma^2 s_i^2 \frac{C(s_{i+1}, t_j) - 2C(s_i, t_j) + C(s_{i-1}, t_j)}{h_s^2} \\ + \quad & rs_i \left(\frac{C(s_{i+1}, t_j) - C(s_{i-1}, t_j)}{2h_s} \right) - rC(s_i, t_j) = 0 \end{aligned} \tag{2}$$

where (see next slide)

Discretizing the Black-Scholes PDE

where

$$h_t = t_{j+1} - t_j = \frac{T}{N}(j+1) - \frac{T}{N}j = \frac{T}{N} \tag{3}$$

denotes the size of the time mesh and

$$h_s = s_{i+1} - s_i = \frac{s_{\max}}{M}(i+1) - \frac{s_{\max}}{M}i = \frac{s_{\max}}{M} \tag{4}$$

denotes the price mesh.

Also, note that we took the liberty of changing the discretization of Delta from forward differences to central differences. (Why?)

Discretizing the PDE: interior

We can rearrange (2) to isolate $C(s_i, t_{j-1})$:

$$\begin{aligned} C(s_i, t_{j-1}) &= \left(1 - \sigma^2 s_i^2 \frac{h_t}{h_s^2} - r h_t\right) C(s_i, t_j) \\ &\quad + \left(\frac{\sigma^2 s_i^2}{2} \frac{h_t}{h_s^2} - \frac{r s_i h_t}{2 h_s}\right) C(s_{i-1}, t_j) \\ &\quad + \left(\frac{\sigma^2 s_i^2}{2} \frac{h_t}{h_s^2} + \frac{r s_i h_t}{2 h_s}\right) C(s_{i+1}, t_j) \\ &= a_i C(s_i, t_j) + l_i C(s_{i-1}, t_j) + u_i C(s_{i+1}, t_j) \end{aligned} \quad (5)$$

where we used the second equality to define a_i, l_i, and u_i.

Note that a_i, l_i, and u_i DO NOT depend on j because the coefficients of the Black-Shoules PDE (1) are constant in time.

Also note that (5) holds for $i = 1, \ldots, M-1$, and when $i = 1$ or $M - 1$, we need to use the top and bottom boundary conditions.

Discretizing the PDE: boundary

Recall that the discrete boundary conditions for our European call are:

1. The payoff at maturity T: $C(s_i, t_N) = C(s_i, T) = (s_i - K)^+ \quad \forall i$
2. "The bottom": $C(s_0, t_j) = C(0, t_j) = 0 \quad \forall j$.
3. "The top": $C(s_M, t_j) = C(s_{\max}, t_j) = s_M - Ke^{-r(t_N - t_j)} \quad \forall j$

Therefore,

$$\begin{aligned} C(s_1, t_{j-1}) &= a_1 C(s_1, t_j) + l_1 C(s_0, t_j) + u_1 C(s_2, t_j) \\ &= a_1 C(s_1, t_j) + u_1 C(s_2, t_j) \qquad (6) \\ C(s_{M-1}, t_{j-1}) &= a_{M-1} C(s_{M-1}, t_j) + l_{M-1} C(s_{M-2}, t_j) + u_{M-1} C(s_M, t_j) \\ &= a_{M-1} C(s_{M-1}, t_j) + l_{M-1} C(s_{M-2}, t_j) \\ &\quad + u_{M-1} \left(s_M - Ke^{-r(t_N - t_j)} \right) \qquad (7) \end{aligned}$$

Discretizing the PDE: Matrix form

Combining (5), (6), (7), and writing them in matrix form, we see that the column vector

$$(C(s_1, t_{j-1}), C(s_2, t_{j-1}), \cdots, C(s_{M-2}, t_{j-1}), C(s_{M-1}, t_{j-1}))^T \quad (8)$$

equals

$$\begin{bmatrix} a_1 & u_1 & & & \\ l_2 & a_2 & u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & l_{M-2} & a_{M-2} & u_{M-2} \\ & & & l_{M-1} & a_{M-1} \end{bmatrix} \begin{bmatrix} C(s_1, t_j) \\ C(s_2, t_j) \\ \vdots \\ C(s_{M-2}, t_j) \\ C(s_{M-1}, t_j) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b_j \end{bmatrix} \quad (9)$$

where

$$b_j = u_{M-1} \left(s_M - K e^{-r(t_N - t_j)} \right) \quad (10)$$

Discretizing the PDE: Matrix form

If we denote the matrix in (9) by A, the first vector by $\vec{C}(j)$, and the second by $\vec{b}(j)$, then we can write the equality (8) and (9) as:

$$\vec{C}(j-1) = A\vec{C}(j) + \vec{b}(j). \tag{11}$$

But (11) is true for all j. Therefore,

$$\begin{aligned} \vec{C}(j-2) &= A\vec{C}(j-1) + \vec{b}(j-1) \\ &= A\left(A\vec{C}(j) + \vec{b}(j)\right) + \vec{b}(j-1) \\ &= A^2\vec{C}(j) + A\vec{b}(j) + \vec{b}(j-1) \end{aligned} \tag{12}$$

and, similarly,

$$\begin{aligned} \vec{C}(j-3) &= A\vec{C}(j-2) + \vec{b}(j-2) \\ &= A^3\vec{C}(j) + A^2\vec{b}(j) + A\vec{b}(j-1) + \vec{b}(j-2). \end{aligned} \tag{13}$$

Discretizing the PDE: Matrix form

Continuing to substitute, we eventually get:

$$\vec{C}(0) = A^j \vec{C}(j) + \sum_{k=0}^{j-1} A^k \vec{b}(k+1). \tag{14}$$

If we use $j = N$ in the formula above, we get

$$\vec{C}(0) = A^N \vec{C}(N) + \sum_{k=0}^{N-1} A^k \vec{b}(k+1). \tag{15}$$

But $\vec{C}(N)$ is the payoff boundary condition, while all $\vec{b}(j)$ come from the "top" boundary. In other words, they are known *a priori*. Therefore, (15) will give us the solution, so long as there are no surprises from A.

We need to make sure that A^N stays sane for large N, e.g., does not grow exponentially or oscillate or misbehave in some other way.

Discretizing the PDE: Stability

Recall that

$$A = \begin{bmatrix} a_1 & u_1 & & & \\ l_2 & a_2 & u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & l_{M-2} & a_{M-2} & u_{M-2} \\ & & & l_{M-1} & a_{M-1} \end{bmatrix} \tag{16}$$

where

$$\begin{aligned} a_i &= 1 - \sigma^2 s_i^2 \frac{h_t}{h_s^2} - r h_t \\ l_i &= \frac{\sigma^2 s_i^2}{2} \frac{h_t}{h_s^2} - \frac{r s_i h_t}{2 h_s} \\ u_i &= \frac{\sigma^2 s_i^2}{2} \frac{h_t}{h_s^2} + \frac{r s_i h_t}{2 h_s} \end{aligned} \tag{17}$$

Discretizing the PDE: Stability

Unfortunately full stability analysis is outside the scope of our course, nor is it needed. We can, however, get an idea of what's involved.

In practice, we would first choose h_s, which controls the precision of our answer, since we know (from theory of parabolic PDEs) that the answer is a nice, smooth function.

Since the matrix A will be raised to high power, we want to make sure its eigenvalues stay below 1. In order for that to happen, we would like

$$|a_i| < 1 \qquad \forall i. \tag{18}$$

This condition is neither necessary nor sufficient, but it does give us a sense of what's involved: the martix A is tridiagonal and its off-diagonal terms tend to be smaller than the diagonal ones, so heuristically (but not in any precise sense), we can think of it as diagonal.

Discretizing the PDE: Stability

The bound (18) will not be true unless

$$\left|\sigma^2 s_{\max}^2 \frac{h_t}{h_s^2} + r h_t\right| < 2 \tag{19}$$

so we would want something stronger, like

$$\left|\sigma^2 s_{\max}^2 \frac{h_t}{h_s^2}\right| < 1 \tag{20}$$

which looks a lot like the CFL condition for numerical PDEs. (Please look it up, it's very important to be aware of it, but we can't get into proving it in this course.)

Condition (20) tells us that we must have

$$h_t < \left(\frac{1}{\sigma^2 s_{\max}^2}\right) h_s^2 = O(h_s^2) \tag{21}$$

which makes for a very fine (i.e., time-consuming) time mesh.

Discretizing the PDE: Explicit and Implicit Schemes

Condition (21) is very inconvenient to obey, because it can force you to have a lot of time steps. Is there any way to avoid it? To find out, let's discretize Theta in (2) using the forward difference:

$$\begin{aligned} & \frac{C(s_i, t_{j+1}) - C(s_i, t_j)}{h_t} \\ + \quad & \frac{1}{2}\sigma^2 s_i^2 \frac{C(s_{i+1}, t_j) - 2C(s_i, t_j) + C(s_{i-1}, t_j)}{h_s^2} \\ + \quad & r s_i \left(\frac{C(s_{i+1}, t_j) - C(s_{i-1}, t_j)}{2h_s} \right) - rC(s_i, t_j) = 0 \end{aligned} \tag{22}$$

Discretizing the PDE: Explicit and Implicit Schemes

Solving for $C(s_i, t_{j+1})$ we get:

$$\begin{aligned} C(s_i, t_{j+1}) &= \left(1 + \sigma^2 s_i^2 \frac{h_t}{h_s^2} + r h_t\right) C(s_i, t_j) \\ &\quad -\frac{1}{2}\left(\sigma^2 s_i^2 \frac{h_t}{h_s^2} - \frac{r s_i h_t}{h_s}\right) C(s_{i-1}, t_j) \\ &\quad -\frac{1}{2}\left(\sigma^2 s_i^2 \frac{h_t}{h_s^2} + \frac{r s_i h_t}{h_s}\right) C(s_{i+1}, t_j) \\ &= d_i C(s_i, t_j) - l_i C(s_{i-1}, t_j) - u_i C(s_{i+1}, t_j) \quad (23) \end{aligned}$$

where d_i is defined by the second equality, while l_i and u_i are the same as in (5).

Discretizing the PDE: Implicit Schemes

In matrix form (23) tells us that the column vector

$$(C(s_1,t_{j+1}),C(s_2,t_{j+1}),\cdots,C(s_{M-2},t_{j+1}),C(s_{M-1},t_{j+1}))^T \quad (24)$$

equals

$$\begin{bmatrix} d_1 & -u_1 & & & \\ -l_2 & d_2 & -u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -l_{M-2} & d_{M-2} & -u_{M-2} \\ & & & -l_{M-1} & d_{M-1} \end{bmatrix} \begin{bmatrix} C(s_1,t_j) \\ C(s_2,t_j) \\ \vdots \\ C(s_{M-2},t_j) \\ C(s_{M-1},t_j) \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ b_j \end{bmatrix} \quad (25)$$

where b_j is the same as in (10). If we denote the matrix in (25) by B, then

$$\vec{C}(j+1) = B\vec{C}(j) - \vec{b}(j). \quad (26)$$

Discretizing the PDE: Implicit Schemes

Now, remember, we can only advance our scheme from larger j to smaller ones, because we have to start at maturity and step towards the present.

Therefore, in the equation (26) the unknown is $\vec{C}(j)$ and to find it we must be able to invert B:

$$\vec{C}(j) = B^{-1}\left(\vec{C}(j+1) + \vec{b}(j)\right). \tag{27}$$

There is theory outside the scope of this course that tells us that B will generally be invertible, certainly far more often than A in (16). Importantly, it is not subject to the CFL condition. In practice, I would check its invertibility with SVD. There are fast algorithms that can invert a tridiagonal matrix in $O(M)$ time and their implementations are widely available.

Discretizing the PDE: Explicit and Implicit Schemes

The first method we saw in (5) is an example of an *explicit* scheme, i.e., where no matrix inversion is required, and the second method (22) is an example of an *implicit* scheme, i.e., when we do need to invert a matrix in order to advance.

In general, implicit schemes are much more stable and are not subject to CFL condition. At your leisure, you should definitely try to violate the CFL condition with an explicit scheme (which includes all tree-based methods, by the way) to see how spectacularly things can blow up.

Official names for the schemes we saw in this lecture are *explicit Euler scheme* and *implicit Euler scheme*.

Discretizing the PDE: Explicit and Implicit Schemes

Let's take another look at the two schemes.

The explicit scheme came from (2):

$$\begin{aligned}\frac{C(s_i,t_j)-C(s_i,t_{j-1})}{h_t} &= -\frac{1}{2}\sigma^2 s_i^2 \frac{C(s_{i+1},t_j)-2C(s_i,t_j)+C(s_{i-1},t_j)}{h_s^2} \\ &\quad -rs_i\left(\frac{C(s_{i+1},t_j)-C(s_{i-1},t_j)}{2h_s}\right) \\ &\quad +rC(s_i,t_j)=F(t_j) \qquad (28)\end{aligned}$$

where we use the last equality to define the function F.

And the implicit scheme comes from this discretization:

$$\frac{C(s_i,t_{j+1})-C(s_i,t_j)}{h_t}=F(t_j) \qquad (29)$$

Discretizing the PDE: Explicit and Implicit Schemes

Let's rewrite (28) and (29) side-by-side after shifting the index in (29) by one:

$$\begin{aligned} \frac{C(s_i, t_j) - C(s_i, t_{j-1})}{h_t} &= F(t_j) \\ \frac{C(s_i, t_j) - C(s_i, t_{j-1})}{h_t} &= F(t_{j-1}) \end{aligned} \tag{30}$$

In other words, we are trying to approximate Theta with right and left values of F on the interval $[t_{j-1}, t_j]$.

But, of course, both schemes are trying to compute $C(s_i, t_j)$, the integral of Theta.

As we had seen in the lectures on quadrature, these look a lot like right and left Riemann rules, which are much worse approximations to the integral than the midpoint or trapezoidal rules.

Discretizing the PDE: Crank-Nicolson

Using the trapezoidal rule as our inspiration, we try the following scheme:

$$\frac{C(s_i, t_j) - C(s_i, t_{j-1})}{h_t} = \frac{F(t_j) + F(t_{j-1})}{2} \tag{31}$$

which, after some algebra, becomes

$$\begin{aligned} &(3 - a_i)C(s_i, t_{j-1}) - u_i C(s_{i+1}, t_{j-1}) - l_i C(s_{i-1}, t_{j-1}) \\ &= (a_i + 1)C(s_i, t_j) + u_i C(s_{i+1}, t_j) + l_i C(s_{i-1}, t_j) \end{aligned} \tag{32}$$

for $i = 1, \ldots, M-1$ with the boundary conditions

$$C(s_0, t_j) = 0 \quad \text{and} \quad C(s_M, t_j) = s_M - Ke^{-r(t_N - t_j)} \quad \forall j \tag{33}$$

Discretizing the PDE: Crank-Nicolson

In matrix form the left hand side of (32) is

$$A_1 \vec{C}(j-1) - \vec{b}(j-1), \tag{34}$$

where $\vec{C}(j-1)$ is defined in (8), $\vec{b}(j-1)$ is defined in (11), and

$$A_1 = \begin{bmatrix} 3-a_1 & -u_1 & & & & \\ -l_2 & 3-a_2 & -u_2 & & & \\ & \ddots & \ddots & \ddots & & \\ & & -l_{M-2} & 3-a_{M-2} & -u_{M-2} & \\ & & & -l_{M-1} & 3-a_{M-1} & \end{bmatrix} \tag{35}$$

Discretizing the PDE: Crank-Nicolson

Similarly, the right hand side of (32) is

$$A_2\vec{C}(j) + \vec{b}(j), \tag{36}$$

where

$$A_2 = \begin{bmatrix} a_1 + 1 & u_1 & & & \\ l_2 & a_2 + 1 & u_2 & & \\ & \ddots & \ddots & \ddots & \\ & & -l_{M-2} & a_{M-2} + 1 & -u_{M-2} \\ & & & -l_{M-1} & a_{M-1} + 1 \end{bmatrix} \tag{37}$$

Discretizing the PDE: Crank-Nicolson

Therefore (32) can be written in matrix form like this:

$$A_1\vec{C}(j-1) - \vec{b}(j-1) = A_2\vec{C}(j) + \vec{b}(j) \tag{38}$$

which can be solved for $\vec{C}(j-1)$ provided A_1 is invertible:

$$\vec{C}(j-1) = A_1^{-1}\left(A_2\vec{C}(j) + \vec{b}(j) + \vec{b}(j-1)\right) \tag{39}$$

This is the famed Crank-Nicholson scheme.

It has been proven that it is stable and is not subject to the CFL condition, just like the fully implicit scheme, therefore allowing for larger h_t. In addition, because it is like the midpoint quadrature rule, it converges faster than the fully implicit scheme.

It is my method of choice when I have to use numerical PDEs.

Discretizing the PDE: Time-dependent coefficients

All the analysis above assumed that the coefficients of (1)

$$\frac{\partial c}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 c}{\partial s^2} + rs\frac{\partial c}{\partial s} - rc = 0 \tag{40}$$

were constant in time.

But often this is not the case: r and σ naturally depend on time, especially if the option we are computing has long expiry.

Discretizing the PDE: Time-dependent coefficients

What changes?

- The matrix A in (11)
- The matrix B in (26)
- Matrices A_1 and A_2 in (35) and (37)

We can no longer precompute them in the beginning of the calculation, but would have to recalculate them at every step.

Luckily they are tridiagonal matrices and therefore can be computed *and inverted* in $O(M)$ operations, which would still keep the PDE scheme manageable.

Discretizing the PDE: Stochastic volatility

But what if the volatility is stochastic as recognized in the Heston model? We will only outline the issues that come up: full analysis is outside the scope of this course.

Recall that the SDE for the Heston model is:

$$\begin{aligned} dS_t &= \mu S_t\,dt + \sigma_t S_t\,dW_t^1 \\ d\sigma_t^2 &= \kappa(\theta - \sigma_t^2)\,dt + \xi\sigma_t\,dW_t^2 \\ Cov(dW_t^1, dW_t^2) &= \rho\,dt \end{aligned} \tag{41}$$

Discretizing the PDE: Stochastic volatility

Before we attempt the discretization of the PDE, we need to see the PDE itself:

$$\begin{aligned}\frac{\partial c}{\partial t} &+ \frac{1}{2}s^2\sigma^2\frac{\partial^2 c}{\partial s^2} + \frac{1}{2}\rho\xi s\sigma\frac{\partial^2 c}{\partial s\partial\sigma} \\ &+ \frac{1}{8}\xi^2\frac{\partial^2 c}{\partial\sigma^2} - \frac{1}{8\sigma}\xi^2\frac{\partial c}{\partial\sigma} \\ &+ rs\frac{\partial c}{\partial s} + \kappa(\theta-\sigma^2)\frac{1}{2\sigma}\frac{\partial c}{\partial\sigma} - rc = 0 \qquad (42)\end{aligned}$$

Discretizing the PDE: Stochastic volatility

- The main thing to notice about the PDE in (42) is that is has two space dimensions:
  - s for the value of the security
  - σ for the volatility.
- Therefore, if we discretize the Heston PDE as before, let's say for Crank-Nicholson, the vector $\vec{C}(j)$ in (39) will be M^2 long, not M as before, which already would slow things down significantly.
- Worse, the matrices A_1 and A_2 will no longer be tridiagonal, because they will depend on nearest neighbor both along the s and the σ coordinates. (Why?)

Discretizing the PDE: Stochastic volatility

- Fortunately there is a way out of the last problem.
- We can advance in half-steps and on each half-step update only s or only σ dimensions, treating the other as constant. That way each update leads to inverstion of one a tridiagonal matrix, which is very fast.
- Luckily, this idea works and is deserving of a name: Alternative Direction Implicit method or ADI for short.
- We are not going into any depth for these higher-dimensional problems: I only wanted to mention this in case you encounter them.
- Actually, there were some recent advances in solving high-dimentional PDEs numerically by using machine learning.