

Advanced Probability and Options, with numerical methods

Part II

Some numerical methods for option pricing

José E. Figueroa-López

Instructor's class notes

Fall 2009

November 19, 2009

Chapter 1

Finite-difference methods

1.1 Introduction

Finite-differences method is a procedure to find a numerical approximation \tilde{u} to the solution u of a well-posed differential equation at the points of a regular grid of the domain. Given that in option pricing, partial differential equations play a key role (e.g. the Black-Scholes equation or the term structure equation), we shall concentrate on this class of equations. Concretely, we consider the following *terminal value problem*:

$$\partial_t u(t, x) + \mu(t, x) \partial_x u(t, x) + a(t, x) \partial_{xx} u(t, x) - r(t, x) u(t, x) = 0, \quad (1.1)$$

$$u(T, x) = \Upsilon(x), \quad (1.2)$$

which is assumed to admit a unique solution $u(t, x) : [0, T] \times (c, d) \rightarrow \mathbb{R}$, for some given domain $(c, d) \subset \mathbb{R}$ and a given function Υ . Specifically, u is assumed to be both continuous on $[0, T] \times (c, d)$ and $C^{1,2}$ on $(0, T) \times (c, d)$ satisfying (1.1) for any $(t, x) \in (0, T) \times (c, d)$ as well as (1.2) for any $x \in (c, d)$. An important instance of (1.1-1.2) is the Black-Scholes equation:

$$\partial_t v(t, s) + rs \partial_s v(t, s) + \frac{\sigma^2 s^2}{2} \partial_{ss} v(t, s) - r v(t, s) = 0, \quad (1.3)$$

$$v(T, s) = \Phi(s), \quad (1.4)$$

on $(t, s) \in [0, T] \times (0, \infty)$, which solution $v(t, s)$ gives the time- t price of a contingent claim on an underlying $\{S_t\}_t$ with maturity T and payoff $\Phi(S_T)$, when the spot price is s . Of course, the terminal condition (1.2) can be replaced by an initial condition of the form $u(0, x) = \Upsilon(x)$. To illustrate the numerical methods in this section, we often use the so-called *heat equation* below:

$$\partial_t u(t, x) - \partial_{xx} u(t, x) = 0, \quad (1.5)$$

$$u(0, x) = \Psi(x). \quad (1.6)$$

The basic idea of finite-difference methods is to approximate the derivatives of the PDE (1.1) by finite-differences. The derivative f' of a function $f : \mathbb{R} \rightarrow \mathbb{R}$ at a point x can be approximated by any of the following three types of differences:

- Forward difference: $\Delta_{h,1} f(x) := \frac{f(x+h) - f(x)}{h}$;

- Backward difference: $\Delta_{h,0}f(x) := \frac{f(x)-f(x-h)}{h}$;
- Centered difference: $\Delta_{h,1/2}f(x) := \frac{f(x+h)-f(x-h)}{2h}$.

In terms of accuracy, both $f'(x) - \Delta_{h,1}f$ and $f'(x) - \Delta_{h,0}f$ are $O(h)$ as $h \rightarrow 0$. That is,

$$\left| \frac{1}{h} (f'(x) - \Delta_{h,1}f(x)) \right|,$$

remains bounded for h small enough. However, centered differences are more accurate since

$$f'(x) - \Delta_{h,1/2}f(x) = O(h^2),$$

as $h \rightarrow 0$. Indeed, assuming that f is smooth enough, Taylor's expansions imply that

$$\begin{aligned} \Delta_{h,1/2}f(x) &= \frac{1}{2h} \left\{ f(x) + f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3) \right\} \\ &\quad - \frac{1}{2h} \left\{ f(x) - f'(x)h + \frac{1}{2}f''(x)h^2 + O(h^3) \right\} \\ &= f'(x) + O(h^2). \end{aligned}$$

Note that, by iterating the finite-difference operators Δ , one can easily construct approximations for higher order derivatives. For instance, the finite-difference approximation of $f''(x)$ using centered differences (with mesh $h/2$) will be:

$$\begin{aligned} f''(x) &\approx \Delta_{h/2,1/2}(\Delta_{h/2,1/2}f) = \Delta_{h/2,1/2} \left(\frac{f(x+h/2) - f(x-h/2)}{h} \right) \\ &= \frac{1}{h} \left(\frac{f(x+h) - f(x)}{h} - \frac{f(x) - f(x-h)}{h} \right) \\ &= \frac{f(x+h) - 2f(x) + f(x-h)}{h^2}. \end{aligned} \tag{1.7}$$

Once the types of finite-difference approximation have been selected, the next step consists of approximating the derivatives of (1.1) at the points of a regular lattice of the solution's domain. For instance, consider the heat equation (1.5) on $\mathbb{R}_+ \times \mathbb{R}$ and the grid points $\mathcal{G}^{\delta t, \delta x} := \{(t_m, x_n)\}_{n \in \mathbb{Z}, m \in \mathbb{N}}$ given by

$$x_n = n\delta x, \quad t_m = m\delta t,$$

where δt and δx are certain mesh parameters determined by the user. Then, applying forward differences in t and centered differences in x , the solution u of (1.5-1.6) will be such that

$$\frac{u(t_{m+1}, x_n) - u(t_m, x_n)}{\delta t} - \frac{u(t_m, x_{n+1}) - 2u(t_m, x_n) + u(t_m, x_{n-1}))}{\delta x^2} \approx 0, \tag{1.8}$$

$$u(0, x_n) = \Psi(x_n). \tag{1.9}$$

The key idea of the finite-difference method is to determine a function $\tilde{u} : \mathcal{G}^{\delta t, \delta x} \rightarrow \mathbb{R}$ that is a solution of the finite-difference equations:

$$\frac{\tilde{u}(t_{m+1}, x_n) - \tilde{u}(t_m, x_n)}{\delta t} - \frac{\tilde{u}(t_m, x_{n+1}) - 2\tilde{u}(t_m, x_n) + \tilde{u}(t_m, x_{n-1}))}{\delta x^2} = 0, \tag{1.10}$$

$$\tilde{u}(0, x_n) = \Psi(x_n), \tag{1.11}$$

for any $n \in \mathbb{Z}$ and $m \geq 1$. The equations (1.10-1.11) indeed determine uniquely such a function \tilde{u} and, given that u restricted to $\mathcal{G}^{\delta t, \delta x}$ approximately solves (1.10), it is expected that

$$u(t_m, x_n) \approx \tilde{u}(t_m, x_n).$$

Let us show how to compute such a solution \tilde{u} . For simplicity, we write $\tilde{u}_n^m := \tilde{u}(t_m, x_n)$ and

$$\frac{\tilde{u}_n^{m+1} - \tilde{u}_n^m}{\delta t} - \frac{\tilde{u}_{n+1}^m - 2\tilde{u}_n^m + \tilde{u}_{n-1}^m}{\delta x^2} = 0, \quad (1.12)$$

$$\tilde{u}_n^0 = \Psi(x_n). \quad (1.13)$$

Solving \tilde{u}_n^{m+1} in (1.12), we have

$$\tilde{u}_n^{m+1} = p_u \tilde{u}_{n+1}^m + p_s \tilde{u}_n^m + p_d \tilde{u}_{n-1}^m, \quad (1.14)$$

for any $m \geq 1$ and $n \in \mathbb{Z}$, where

$$p_u := p_d := \frac{\delta t}{\delta x^2}, \quad p_s := 1 - 2\frac{\delta t}{\delta x^2}.$$

Note that (1.14) allows, in principle, to compute \tilde{u}_n^m for any $n \in \mathbb{Z}$, $m \geq 1$ in terms of the initial values (1.13). For instance, suppose that $p_u = p_d = 1/4$ and $p_s = 1/2$ and we want to find \tilde{u}_1^2 . Then,

$$\begin{aligned} \tilde{u}_1^2 &= \frac{1}{4}\tilde{u}_2^1 + \frac{1}{2}\tilde{u}_1^1 + \frac{1}{4}\tilde{u}_0^1 \\ &= \frac{1}{16}\tilde{u}_3^0 + \frac{1}{8}\tilde{u}_2^0 + \frac{1}{16}\tilde{u}_1^0 + \frac{1}{8}\tilde{u}_2^0 + \frac{1}{4}\tilde{u}_1^0 + \frac{1}{8}\tilde{u}_0^0 + \frac{1}{16}\tilde{u}_1^0 + \frac{1}{8}\tilde{u}_0^0 + \frac{1}{16}\tilde{u}_{-1}^0. \end{aligned}$$

In practice, to implement (1.14), we need to restrict our lattice $\mathcal{G}^{\delta t, \delta x}$ and maybe imposed some additional boundary conditions. For instance, if we are only interested in finding the solution u for $t = T$ and $x = x_0$, we can set $\delta t = T/M$, for some large M , and take a triangular-shaped lattice

$$t_m = m\delta t, \quad x_n = x_0 + n\delta x, \quad m = 0, \dots, M, \quad n = -(N - m), \dots, (N - m),$$

with a small mesh δx and a large enough N . In fact, taking $N \geq M$ will suffice to determine $\tilde{u}(T, x_0)$ uniquely from the values of \tilde{u} at $t = 0$.

In some cases, we can infer some information regarding the behavior of $u(t, x)$ (or \tilde{u}) when $x \rightarrow \infty$ or $x \rightarrow -\infty$. For instance, if $\Psi(x) = 0$ for $x \geq K$ and $x \leq 0$, then we expect that $\tilde{u}(t_m, x_n) \equiv 0$ for x_n large enough positive or negative. In that case, one can enforce this condition on \tilde{u} and set

$$\tilde{u}(t_m, x_N) = \tilde{u}(t_m, x_{-N}) = 0.$$

The above fictitious boundary condition will allow us to compute \tilde{u} for any $m = 1, \dots, M$ and $n = -(N - 1), \dots, N - 1$.

There are two common types of boundary conditions:

- **Dirichlet type:** e.g. $u(0, x) = \Psi(x)$, $u(t, \infty) = \beta(t)$, $u(t, -\infty) = \alpha(t)$, for some known functions Ψ , α , and β .

- Neuman type: e.g. $\partial_x u(t, \infty) = \dot{\beta}(t)$, $\partial_x u(t, -\infty) = \dot{\alpha}(t)$, for some known functions $\dot{\alpha}$, and $\dot{\beta}$.

Neuman type conditions will translate into a finite-difference equation for \tilde{u} at the boundary points. For instance, $\partial_x u(t, \infty) = \dot{\beta}(t)$ can be taken into account by imposing the condition:

$$\tilde{u}(t_m, x_N) = \tilde{u}(t_m, x_{N-1}) + \dot{\beta}(t_m)\delta x,$$

for any $m = 1, \dots, M$.

In summary, the general steps to find a finite-difference approximation \tilde{u} for (1.1-1.2) are as follows:

- (1) Discretize time and space on a region of interest, say $[t_0, t_N] \times [x_{-N}, x_N]$, leading to a lattice $\{(t_m, x_n)\}$ determined by given mesh parameters δt and δx as follows:

$$x_n = x_0 + n\delta x, \quad t_m = t_0 + m\delta t, \quad n = -N, \dots, N, \quad m = 0, \dots, M.$$

- (2) Approximate the derivatives of the PDE (1.1) at each point of the lattice by some type finite difference. This discretization process induces a *system of finite-difference equations* that the approximating function $\tilde{u}(t_m, x_n)$ have to satisfy.
- (3) Impose boundary conditions that, together with the finite-difference equations of step (2), determines uniquely $\tilde{u}(t_m, x_n)$ in the lattice $\{(t_m, x_n)\}$.
- (4) Solve the system of finite-differences with the boundary conditions on the lattice points.

1.2 The explicit method

The explicit method own its name to the fact that the one can explicitly solve the system of finite-differences equations resulting from the discretization of the differential equation. The approximation scheme (1.12-1.13) for the heat equation (1.5-1.6) is an instance of this method. Indeed, one can explicitly find \tilde{u} applying (1.14) iteratively for each time step $m = 2, \dots, M$.

In the case of a terminal value problem like (1.1-1.2) on $[0, T] \times \mathbb{R}$, the explicit method will work backwards in time. Concretely, let

$$t_m = m\delta t, \quad x_n = n\delta x, \quad m = 0, \dots, M, \quad n \in \mathbb{Z},$$

where $\delta t = T/M$ for some $M \geq 1$ and $\delta x > 0$. Then, applying backward difference approximations in time and centered differences in space to (1.1), it follows that $u_n^m := u(t_m, x_n)$ approximately satisfies the equations

$$\frac{\tilde{u}_n^m - \tilde{u}_n^{m-1}}{\delta t} + \mu_n^m \frac{\tilde{u}_{n+1}^m - \tilde{u}_{n-1}^m}{\delta x} + a_n^m \frac{\tilde{u}_{n+1}^m - 2\tilde{u}_n^m + \tilde{u}_{n-1}^m}{(\delta x)^2} - r_n^m \tilde{u}_n^m = 0, \quad (1.15)$$

$$\tilde{u}_n^M = \Upsilon(x_n), \quad (1.16)$$

where $\mu_n^m := \mu(t_m, x_n)$, $a_n^m := a(t_m, x_n)$, and $r_n^m := r(t_m, x_n)$. Hence, under certain conditions, it is expected that the solution \tilde{u}_n^m of (1.15-1.33) is such that

$$u(t_m, x_n) \approx \tilde{u}_n^m,$$

for all $m = 1, \dots, M$ and $n \in \mathbb{Z}$. Solving (1.15), \tilde{u}_n^{m-1} can be computed explicitly (in terms of \tilde{u}^m) as follows:

$$\tilde{u}_n^{m-1} = p_u^{n,m} \cdot \tilde{u}_{n+1}^m + p_s^{n,m} \cdot \tilde{u}_n^m + p_d^{n,m} \cdot \tilde{u}_{n-1}^m, \quad (1.17)$$

where

$$p_u^{n,m} = \frac{a_n^m \delta t}{(\delta x)^2} + \frac{\mu_n^m \delta t}{2(\delta x)}, \quad p_d^{n,m} = \frac{a_n^m \delta t}{(\delta x)^2} - \frac{\mu_n^m \delta t}{2(\delta x)}, \quad p_s^{n,m} = 1 - (\delta t)r_n^m - p_d^{n,m} - p_u^{n,m}.$$

In principle, one could apply the previous scheme to solve the Black-Scholes equation (1.3-1.4). However, for simplicity and stability reasons, it is better to transform this equation into a PDE with constant coefficients. Concretely, it is quite easy to check that v is of the form

$$v(t, s) = e^{rt} u(t, \ln s), \quad (1.18)$$

where $u(t, x)$ solves the problem

$$\partial_t u(t, x) + \left(r - \frac{\sigma^2}{2}\right) \partial_x u(t, x) + \frac{\sigma^2}{2} \partial_{xx} u(t, x) = 0, \quad (1.19)$$

$$u(T, x) = e^{-rT} \Upsilon(e^x). \quad (1.20)$$

Financially, $u(t, x) = e^{-rt} v(t, e^x)$ can be interpreted as the present value of the time- t price of the T -claim with payoff $\Phi(S_T)$ when the spot log return $\log S_t = x$.

The explicit method can then be applied to (1.19), which solution can be obtained backwards applying

$$\tilde{u}_n^{m-1} = p_u \cdot \tilde{u}_{n+1}^m + p_s \cdot \tilde{u}_n^m + p_d \cdot \tilde{u}_{n-1}^m, \quad (1.21)$$

with

$$p_u = \frac{\sigma^2 \delta t}{2(\delta x)^2} + \frac{(r - \frac{\sigma^2}{2}) \delta t}{2(\delta x)}, \quad p_d = \frac{\sigma^2 \delta t}{2(\delta x)^2} - \frac{(r - \frac{\sigma^2}{2}) \delta t}{2(\delta x)}, \quad p_s = 1 - p_d - p_u.$$

When δx is small enough, $p_u, p_d, p_s \geq 0$ and they can be interpreted as probability weights so that \tilde{u}_n^{m-1} is a weighted average of the three nodes in front of it. This procedure will work as a backward induction procedure applied to a trinomial tree approximation of $\ln S_t$.

The following is the (MATLAB) pseudo code of the explicit method to find the time 0 price of a T -claim $\mathcal{X} = \Phi(S_T)$ when the spot stock price is $S_0 = s$. Note that the code below does not require boundary conditions

```

1. % Initialization
2.   deltat = T/M;
3.   N = M; % Or any other integer N larger than M
4.   t = (0:1:M)*deltat;
5.   x = ln(s) + (-N:1:N)*deltax;
6.   u=zeros(2N+1,M+1);
7.   for n=1:2N+1
8.       u(n,M+1) = Phi(exp(x(n)));
9.   end;
10. % Computation of probabilities
11.   p_u = ... ;

```

```

12.     p_d = ... ;
13.     p_s = 1-p_u-p_d;
14. % Solving the system of finite-differences
15.     for m=M:-1:1 % We go backward in time
16.         for n=M-m+2:1:M+m % We go upward in space
17.             u(n,m)=p_u * u(n+1,m+1)+p_s * u(n,m+1) +p_d * u(n-1,m+1);
18.         end
19.     end
20. % Computation of the price;
21.     ClaimPrice = u(1,M+1);

```

In the case of a vanilla option, one can impose a boundary condition for x large positive or negative. For instance, for a call option $\mathcal{X} = (S_T - K)_+$, it follows that the solution v of (1.3-1.4) is such that

$$\lim_{s \rightarrow \infty} \partial_s v(t, s) = 1, \quad \lim_{s \rightarrow 0} \partial_s v(t, s) = 0.$$

Since $\partial_x u(t, x) = e^{-rt} e^x \partial_s v(t, e^x)$, it follows that one can impose the following boundary conditions for each time step $m = 1, \dots, M - 1$, when x_N is large positive and x_{-N} is large negative:

$$\tilde{u}_{-N}^m = \tilde{u}_{-N+1}^m, \quad \text{and} \quad \tilde{u}_N^m = \tilde{u}_{N-1}^m + (\delta x) e^{-rt_m + x_N} \approx \tilde{u}_N^m + e^{-rt_m} (e^{x_N} - e^{x_{N-1}}).$$

In terms of the above pseudo-code, we need to replace lines 15-19 as follows:

```

13. % Solving the system of finite-differences
14.     for m=M:-1:1 % We go backward in time
15.         for n=2:1:2N % We go upward in space
16.             u(n,m)=p_u * u(n+1,m+1)+p_s * u(n,m+1) +p_d * u(n-1,m+1);
17.         end
18.         % Boundary conditions
19.         u(1,m)=u(2,m);
20.         u(2N+1,m)=u(2N,m)+exp(-r*t(m)) * (exp(x(2N+1))-exp(x(2N)));
21.     end
22. % Computation of the price;
23.     ClaimPrice = u(1,M+1);

```

1.3 Convergence, stability, and consistency

The usefulness of a numerical approximation method is judged based on its efficiency, accuracy, and stability. Efficiency refers to how fast the method is to attain certain degree of accuracy. In principle, it is expected that finite difference methods will be more accurate if the mesh parameters δt and δx become smaller. We shall see that this is not necessarily the case and that certain stability relationship between δt and δx must hold. In general terms, stability is the property of a numerical method to avoid uncontrolled error propagation whether the error is in the input data or roundoff error. One of the most fundamental results of numerical analysis, *the LAX Equivalence Theorem*, establish a close connection between the stability and convergence of finite-difference methods.

We need some terminology. Let L be the differential operator associated with an initial or final value problem. For instance, the differential operator associate with the initial value problem (1.5-1.6) is the function L that maps a function $u : [0, \infty) \times \mathbb{R} \rightarrow \mathbb{R}$ that is $C^{1,2}$ on $(0, \infty) \times \mathbb{R}$ and continuous on $[0, \infty) \times \mathbb{R}$ into the function $U : [0, \infty) \times \mathbb{R} \rightarrow \mathbb{R}$ defined by

$$U(t, x) = \frac{\partial u}{\partial t}(t, x) - \frac{\partial^2 u}{\partial x^2}(t, x).$$

We can write

$$L = \frac{\partial}{\partial t} - \frac{\partial^2}{\partial x^2}. \quad (1.22)$$

Let $L^{\delta t, \delta x}$ be a "finite-difference" operator on a lattice $\mathcal{G}^{\delta t, \delta x} := \{(t_m, x_n)\}_{n \in \mathbb{Z}, m \in \mathbb{N}}$, where $t_m = m\delta t$ and $x_n = n\delta x$. Concretely, $L^{\delta t, \delta x}$ maps arrays $\tilde{u} := [\tilde{u}_n^m]_{n \in \mathbb{Z}, m \in \mathbb{N}}$ into arrays $\tilde{U} := [\tilde{U}_n^m]_{n \in \mathbb{Z}, m \in \mathbb{N}}$. Typically, the operator $L^{\delta t, \delta x}$ will be motivated by a finite-difference discretization methods. In the case of the explicit method (1.12)-(1.13), one can consider the finite-difference operator $L_{\delta t, \delta x}$ defined as follows:

$$\tilde{U}_n^m := [L^{\delta t, \delta x} \tilde{u}]_n^m := \frac{\tilde{u}_n^{m+1} - \tilde{u}_n^m}{\delta t} - \frac{\tilde{u}_{n+1}^m - 2\tilde{u}_n^m + \tilde{u}_{n-1}^m}{\delta x^2}, \quad (1.23)$$

for any $n \in \mathbb{Z}$ and $m \in \mathbb{N}$. We say that a finite-difference operator $L^{\delta t, \delta x}$ is consistent for a differential operator L if for any function u in the domain of L and any bounded domain $D = [0, T] \times [c, d]$,

$$\lim_{\delta t, \delta x \rightarrow 0} \sup_{(t_m, x_n) \in D} |[L^{\delta t, \delta x} \tilde{u}]_n^m - Lu(t_m, x_n)| = 0,$$

where above \tilde{u} is the array $\tilde{u}_n^m := u(t_m, x_n)$.

Example 1. Note that (1.23) is a consistent finite difference operator for the heat differential operator (1.22). Furthermore, by Taylor's expansions, one can show that

$$[L^{\delta t, \delta x} \tilde{u}]_n^m - Lu(t_m, x_n) = O(\delta t) + O((\delta x)^2),$$

and we say that the discretization error of $L^{\delta t, \delta x}$ is $O(\delta t) + O((\delta x)^2)$.

Let $[\tilde{u}^{\delta t, \delta x}]_n^m$ denote the solution of the following finite-difference initial problem:

$$L^{\delta t, \delta x} \tilde{u} = 0, \quad \tilde{u}_n^0 = \Phi(x_n) \quad (1.24)$$

for a function Φ . We say that $L^{\delta t, \delta x}$ is stable if for any bounded domain $D = [0, T] \times [c, d]$, there exists a constant $K < \infty$ such that

$$\max \{ |[\tilde{u}^{\delta t, \delta x}]_n^m| : (t_m, x_n) \in D \} \leq K,$$

for any $\delta t > 0$ and $\delta x > 0$ small enough.

The following fundamental result, called *LAX Equivalence Theorem*, provides necessary and sufficient conditions for convergence.

Theorem 1. Let L be a differential operator defining a well-posed initial value problem

$$Lu(t, x) = 0, \quad u(0, x) = \Phi(x),$$

and let $L^{\delta t, \delta x}$ be a consistent finite-difference operator for L . Let $\tilde{u}^{\delta t, \delta x}$ be the solution of the system (1.24). Then,

$$\lim_{\delta t, \delta x \rightarrow 0} \sup_{(t_m, x_n) \in D} |[\tilde{u}^{\delta t, \delta x}]_n^m - u(t_m, x_n)| = 0,$$

for any bounded domain $D = [0, T] \times [c, d]$ if and only if $L^{\delta t, \delta x}$ is stable.

The usefulness of the previous result stems from the fact that stability is typically easier to check than convergence. The following example shows that

$$\alpha := \frac{\delta t}{(\delta x)^2} \leq \frac{1}{2}. \quad (1.25)$$

is a sufficient condition for the finite-difference operator of (1.23) to be stable. Given that the solutions of (1.24) coincides with those of the explicit method, in light of LAX theorem, the same conclusion will hold for the explicit method applied to the heat equation.

Example 2. Let \tilde{u} be the solution of (1.24) with initial condition

$$\tilde{u}_n^0 = e^{ikx_n} = \cos(kx_n) + i \sin(kx_n),$$

for a constant k . Using the explicit solution (1.14), we have that

$$\begin{aligned} \tilde{u}_n^1 &= \alpha \tilde{u}_{n+1}^0 + (1 - 2\alpha) \tilde{u}_n^0 + \alpha \tilde{u}_{n-1}^0 = \alpha e^{ikx_{n+1}} + (1 - 2\alpha) e^{ikx_n} + \alpha e^{ikx_{n-1}} \\ &= e^{ikx_n} \left\{ 1 + \frac{\delta t}{(\delta x)^2} [e^{ik\delta x} + e^{-ik\delta x} - 2] \right\} = \tilde{u}_n^0 \left\{ 1 + \frac{2\delta t}{(\delta x)^2} [\cos(k\delta x) - 1] \right\} \end{aligned}$$

By induction, it follows that

$$\tilde{u}_n^m = \tilde{u}_n^0 \left\{ 1 + \frac{2\delta t}{(\delta x)^2} [\cos(k\delta x) - 1] \right\}^m.$$

It is now clear that, for the solutions \tilde{u} to remain bounded, it suffices that

$$\left| 1 + \frac{2\delta t}{(\delta x)^2} [\cos(k\delta x) - 1] \right| \leq 1,$$

which can be guaranteed if (1.25) holds true since, in that case,

$$-1 \leq 1 - \frac{4\delta t}{(\delta x)^2} \leq 1 + \frac{2\delta t}{(\delta x)^2} [\cos(k\delta x) - 1] \leq 1.$$

It is natural to ask to what extend (1.25) is also necessary for a general initial condition $\Phi(x)$. In order to answer this, let us assume that Φ admits a Fourier series representation:

$$\Phi(x) = \sum_{k=-\infty}^{\infty} c_k e^{ikx}.$$

The class of functions with this representation is known to be quite large. Now, it can be shown by induction that the solution \tilde{u}_n^m has the representation

$$\tilde{u}_n^m = \sum_{k=-\infty}^{\infty} c_k e^{ikx_n} \left\{ 1 + \frac{2\delta t}{(\delta x)^2} [\cos(k\delta x) - 1] \right\}^m.$$

The above expression suggests that no matter how small δx is, one cannot rule out the possibility that $\cos(k\delta x) - 1$ is close to its worst case value -2 for a large enough k^* . Hence, when $\alpha > 1/2$, the term corresponding to that k^* will be quite large when m is large, likely leading to unstable solutions.

For financial applications, the stability condition (1.25) seems to be essential as the following example taken from Wilmott, Howison, and Dewynne shows:

Example 3. W, H, & D (pp. 142-143) apply the explicit method to the heat equation, and from this, recover the value of a put option with strike $K = 10$, and maturity $T = 1/2$ year on a underlying following the Black-Scholes model with volatility $\sigma = 20\%$ and risk-free interest rate $r = 5\%$. The following table extracts some results for different initial spot prices S_0 and different values of α .

S_0	$\alpha = .25$	$\alpha = .5$	$\alpha = .52$	Exact
7	2.75	2.75	-17.41	2.75
10	.44	.44	625	.44
14	.0028	.0027	-15.21	.0028

Even with $\alpha = .52$, there are serious numerical problems especially for near-at-the-money options, which are the most liquid in the market.

1.4 Implicit methods

The inherent stability condition $\delta t / (\delta x)^2 < 1$ of the explicit methods implies that if one wishes to improve the accuracy of the method by doubling the number of x -points, one must quarter the time mesh, taking 8 times longer. At the expense of a loss in speed, implicit methods will allow us to increase the number of x -points without having to take very small time mesh. The general idea consists of approximating the derivative in time by a linear combination of forward and backward difference approximations as follows:

$$\frac{\partial u(t, x)}{\partial t} \approx (1 - \theta) \cdot \frac{u(t + \delta t, x) - u(t, x)}{\delta t} + \theta \cdot \frac{u(t, x) - u(t - \delta t, x)}{\delta t}, \quad (1.26)$$

The most common methods of this kind are when $\theta = 1$, known as the fully implicit method, and when $\theta = 1/2$, known as the Crank-Nicolson method. We now describe the system of finite-difference equations corresponding to these two methods.

1.4.1 Fully implicit method

Again, as a way of an example, let us consider the heat problem (1.5-1.6). The explicit method (1.12) results from applying forward differences in time and centered differences in space. In the

fully implicit method, we consider instead backward difference in time. Concretely, given a regular lattice $\mathcal{G}^{\delta t, \delta x} := \{(t_m, x_n)\}_{n \in \mathbb{Z}, m \in \mathbb{N}}$, the solution u of (1.5-1.6) will be such that

$$\frac{u(t_m, x_n) - u(t_{m-1}, x_n)}{\delta t} - \frac{u(t_m, x_{n+1}) - 2u(t_m, x_n) + u(t_m, x_{n-1}))}{\delta x^2} \approx 0, \quad (1.27)$$

for $m = 1, 2, \dots$ and $n \in \mathbb{Z}$. The previous condition on $\{u(t_m, x_n)\}$ motivates to consider the solution of the finite-difference equations:

$$\frac{\tilde{u}_n^m - \tilde{u}_n^{m-1}}{\delta t} - \frac{\tilde{u}_{n+1}^m - 2\tilde{u}_n^m + \tilde{u}_{n-1}^m}{\delta x^2} = 0, \quad (1.28)$$

$$\tilde{u}_n^0 = \Psi(x_n), \quad (1.29)$$

for any $n \in \mathbb{Z}$ and $m = 1, 2, \dots$. If we were able to find $\{\tilde{u}_n^m\}_{m \in \mathbb{Z}, n \in \mathbb{N}}$ that solves (1.28-1.29), one will expect that

$$u(t_m, x_n) \approx \tilde{u}_n^m.$$

However, unlike the explicit method, one cannot solve for \tilde{u}_n^m explicitly from only the initial values \tilde{u}_n^0 , without imposing some boundary conditions for large N and $-N$. Of course, in order to reduce any error, the boundary conditions should be consistent with the conditions of $u(t, x)$ for large positive x and large negative x . For simplicity, let us assume Dirichlet boundary conditions of the form:

$$\tilde{u}_{-N}^m = 0, \quad \text{and} \quad \tilde{u}_N^m = 0, \quad \text{for} \quad m \geq 1. \quad (1.30)$$

At each time step t_m , $m = 1, 2, \dots$, the equations (1.28-1.30) determine uniquely the vector $\tilde{\mathbf{u}}^m := [\tilde{u}_{-N+1}^m, \dots, \tilde{u}_{N-1}^m]^T$ as a solution of a system of equations of the form:

$$(\text{Id} - \delta t A_{\delta x}) \tilde{\mathbf{u}}^m = \tilde{\mathbf{u}}^{m-1}, \quad (1.31)$$

where Id is the indicator matrix of size $2N - 1$ and $A_{\delta x}$ is the matrix

$$A_{\delta x} := \begin{bmatrix} -\frac{2}{\delta x^2} & \frac{1}{\delta x^2} & 0 & 0 & \dots & \dots & 0 \\ \frac{1}{\delta x^2} & -\frac{2}{\delta x^2} & \frac{1}{\delta x^2} & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 & \frac{1}{\delta x^2} & -\frac{2}{\delta x^2} \end{bmatrix}.$$

We remark that the solution $\tilde{\mathbf{u}}^m$ always exists since the matrix $C := (\text{Id} - \delta t A_{\delta x})$ is always invertible. Indeed, fixing $\alpha = \delta t / (\delta x)^2$,

$$C = [c_{i,j}] := \begin{bmatrix} 1 + 2\alpha & -\alpha & 0 & 0 & \dots & \dots & 0 \\ -\alpha & 1 + 2\alpha & -\alpha & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 & -\alpha & 1 + 2\alpha \end{bmatrix},$$

which is a diagonally dominant (namely, $\sum_{j \neq i} |c_{i,j}| \leq c_{i,i}$, for any row i) and hence, it is invertible.

In the case of the general final value problem (1.1-1.2), the implicit method will result from applying forward differences in time as follows:

$$\frac{\tilde{u}_n^{m+1} - \tilde{u}_n^m}{\delta t} + \mu_n^m \frac{\tilde{u}_{n+1}^m - \tilde{u}_{n-1}^m}{2\delta x} + a_n^m \frac{\tilde{u}_{n+1}^m - 2\tilde{u}_n^m + \tilde{u}_{n-1}^m}{(\delta x)^2} - r_n^m \tilde{u}_n^m = 0, \quad (1.32)$$

where $\mu_n^m := \mu(t_m, x_n)$, $a_n^m := a(t_m, x_n)$, and $r_n^m := r(t_m, x_n)$. The boundary condition now takes the form:

$$\tilde{u}_n^M = \Upsilon(x_n), \quad (1.33)$$

and we will have to work backwards in m to determine $\tilde{\mathbf{u}}^m$. Imposing the Dirichlet boundary conditions (1.30), the implicit method will proceed to solve for $\tilde{\mathbf{u}}^m := [\tilde{u}_{-N+1}^m, \dots, \tilde{u}_{N-1}^m]^T$ the following system:

$$(\text{Id} - \delta t A_{\delta x}^m) \tilde{\mathbf{u}}^m = \tilde{\mathbf{u}}^{m+1}, \quad (1.34)$$

for $m = M - 1, \dots, 0$, where now

$$A_{\delta x}^m = \begin{bmatrix} \beta_{-N+1}^m & \gamma_{-N+1}^m & 0 & 0 & \dots & \dots & 0 \\ \eta_{-N+2}^m & \beta_{-N+2}^m & \gamma_{-N+2}^m & 0 & \dots & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & 0 & \eta_{N-1}^m & \beta_{N-1}^m \end{bmatrix},$$

with

$$\eta_n^m = \frac{a_n^m}{(\delta x)^2} - \frac{\mu_n^m}{2(\delta x)}, \quad \gamma_n^m = \frac{a_n^m}{(\delta x)^2} + \frac{\mu_n^m}{2(\delta x)}, \quad \beta_n^m = -2 \frac{a_n^m}{(\delta x)^2} - r_n^m. \quad (1.35)$$

Note that the tridiagonal matrix $C = (\text{Id} - \delta t \cdot A_{\delta x}^m)$ is not necessarily invertible anymore. However, it is typically not hard to find conditions for C to be diagonally dominant, and hence, invertible.

The solution of the systems (1.31) or (1.34) are typically performed by two methods: LU decomposition of the matrix C or iterative methods. We now proceed to show the iterative method.

1.4.2 Solutions by iterative methods

As seen in the previous section, the implicit method involves solving a tridiagonal system of equations such as (1.34). This kind of problems is well-studied in the literature of numerical methods. In this part we review some iterative methods to find its solution, which are known to be particularly fast and easy to implement. The idea behind these methods comes from the following simple manipulation of (1.32),

$$\tilde{u}_n^m = \frac{1}{1 - (\delta t)\beta_n^m} \left\{ \tilde{u}_n^{m+1} + (\delta t)\eta_n^m \tilde{u}_{n-1}^m + (\delta t)\gamma_n^m \tilde{u}_{n+1}^m \right\}, \quad n = -N + 1, \dots, N - 1, \quad (1.36)$$

where η_n^m , β_n^m , and γ_n^m are as in (1.35). In the case of the heat equation (1.5-1.6), the equation of the implicit method (1.28) will take the form:

$$\tilde{u}_n^m = \frac{1}{1 + 2\alpha} \left\{ \tilde{u}_n^{m-1} + \alpha \tilde{u}_{n-1}^m + \alpha \tilde{u}_{n+1}^m \right\}, \quad n = -N + 1, \dots, N - 1, \quad (1.37)$$

In light of equation (1.36), its solution $\tilde{\mathbf{u}}^m := [\tilde{u}_{-N+1}^m, \dots, \tilde{u}_{N-1}^m]^T$ can be interpreted as the fixed point of certain mapping $H : \mathbb{R}^{2N-1} \rightarrow \mathbb{R}^{2N-1}$. By definition $\tilde{\mathbf{u}}^m$ is said to be the fixed point of $H : \mathbb{R}^{2N-1} \rightarrow \mathbb{R}^{2N-1}$ if $H(\tilde{\mathbf{u}}^m) = \tilde{\mathbf{u}}^m$. Under certain conditions on the mapping H , it is known that the sequence $\{\tilde{\mathbf{u}}^{m,k}\}_{k \geq 0}$ of vectors defined recursive as

$$\tilde{\mathbf{u}}^{m,k+1} = H(\tilde{\mathbf{u}}^{m,k}), \quad k \geq 1$$

converges to the fixed point of the mapping H . Above, $\tilde{\mathbf{u}}^{m,0}$ is an arbitrary initial guess.

The *Jacobi method* applies the previous idea. Concretely, starting with $\tilde{u}_n^{m,0} = \tilde{u}_n^{m+1}$, the method computes recursively the following values

$$\tilde{u}_n^{m,k+1} = \frac{1}{1 + (\delta t)\beta_n^m} \left\{ \tilde{u}_n^{m+1} + \eta_n^m(\delta t)\tilde{u}_{n-1}^{m,k} + \gamma_n^m(\delta t)\tilde{u}_{n+1}^{m,k} \right\},$$

from $n = N - 1$ to $n = -N + 1$.

A simple variation of the previous method, called *Gauss-Seidel method*, consists of using the updated values $\tilde{u}_n^{m,k+1}$ as soon as they become available. Hence, given that the values $\tilde{u}_i^{m,k+1}$ for $i = N - 1, \dots, n + 1$ have already been computed, the subsequent value, $\tilde{u}_n^{m,k+1}$, is computed as follows:

$$\tilde{u}_n^{m,k+1} = \frac{1}{1 + (\delta t)\beta_n^m} \left\{ \tilde{u}_n^{m+1} + \eta_n^m(\delta t)\tilde{u}_{n+1}^{m,k+1} + \gamma_n^m(\delta t)\tilde{u}_{n-1}^{m,k} \right\}.$$

Successive over-relaxation (SOR) is another variation where $\tilde{u}_n^{m,k+1}$ is computed using the following two steps:

$$\begin{aligned} y_n^{m,k+1} &= \frac{1}{1 + (\delta t)\beta_n^m} \left\{ \tilde{u}_n^{m+1} + \eta_n^m(\delta t)\tilde{u}_{n-1}^{m,k+1} + \gamma_n^m(\delta t)\tilde{u}_{n+1}^{m,k} \right\}, \\ \tilde{u}_n^{m,k+1} &= \tilde{u}_n^{m,k} + \omega (y_n^{m,k+1} - \tilde{u}_n^{m,k}), \end{aligned}$$

where $1 < \omega < 2$ (the over-relaxation parameter). It is known that SOR converges whenever $\omega \in (0, 2)$ and furthermore, there is an optimal $\omega^* \in (1, 2)$, that make the method to converge the fastest.