4

FD Methods for Parabolic PDEs

A linear PDE of the form

$$u_t = Lu, \tag{4.1}$$

where t usually denotes the time and L is a linear elliptic differential operator in one or more spatial variables, is called parabolic. Furthermore, the second-order canonical form

$$a(x,t)u_{tt} + 2b(x,t)u_{xt} + c(x,t)u_{xx} + \text{lower-order terms} = f(x,t)$$

is parabolic if $b^2 - ac \equiv 0$ in the entire $x - t$ domain. Note that, we can transform this second-order PDE into a system of two PDEs by setting $v = u_t$, where the t-derivative is first order. Some important parabolic PDE are as follows.

- 1D heat equation with a source

  $$u_t = u_{xx} + f(x,t).$$

  The dimension refers to the space variable (x direction).
- General heat equation

  $$u_t = \nabla \cdot (\beta \nabla u) + f(\mathbf{x},t)\,, \tag{4.2}$$

  where β is the diffusion coefficient and $f(\mathbf{x},t)$ is the source (or sink) term.
- Diffusion-advection equation

  $$u_t = \nabla \cdot (\beta \nabla u) + \mathbf{w} \cdot \nabla u + f(\mathbf{x},t)\,,$$

  where $\nabla \cdot (\beta \nabla u)$ is the diffusion term and $\mathbf{w} \cdot \nabla u$ the advection term.
- Canonical form of diffusion-reaction equation

  $$u_t = \nabla \cdot (\beta \nabla u) + f(\mathbf{x},t,u).$$

  The nonlinear source term $f(\mathbf{x},t,u)$ is a reaction term.

The *steady-state solutions* (when $u_t = 0$) are the solutions of the corresponding elliptic PDEs, *i.e.*,

$$\nabla \cdot (\beta \nabla u) + \bar{f}(\mathbf{x}, u) = 0$$

for the last case, assuming $\lim_{t\to\infty} f(\mathbf{x}, t, u) = \bar{f}(\mathbf{x}, u)$ exists.

Initial and Boundary Conditions

In time-dependent problems, there is an initial condition that is usually specified at $t = 0$, *i.e.*, $u(\mathbf{x}, 0) = u_0(\mathbf{x})$ for the above PDE, in addition to relevant boundary conditions. If the initial condition is given at $t = T \neq 0$, it can of course be rendered at $t = 0$ by a translation $t' = t - T$. Thus for the 1D heat equation $u_t = u_{xx}$ on $a < x < b$ for example, we expect to have an initial condition at $t = 0$ in addition to boundary conditions at $x = a$ and $x = b$ say. Note that the boundary conditions at $t = 0$ may or may not be consistent with the initial condition, *e.g.*, if a Dirichlet boundary condition is prescribed at $x = a$ and $x = b$ such that $u(a, t) = g_1(t)$ and $u(b, t) = g_2(t)$, then $u_0(a) = g_1(0)$ and $u_0(b) = g_2(0)$ for consistency.

Dynamical Stability

The fundamental solution $u(x.t) = e^{-x^2/4t}/\sqrt{4\pi t}$ for the 1D heat equation $u_t = u_{xx}$ is uniformly bounded. However, for the backward heat equation $u_t = -u_{xx}$, if $u(x, 0) \neq 0$ then $\lim_{t\to\infty} u(x, t) = \infty$. The solution is said to be dynamically *unstable* if it is not uniformly bounded, *i.e.*, if there is no constant $C > 0$ such that $|u(x, t)| \leq C$. Some applications are dynamically unstable and "blow up," but we do not discuss how to solve such dynamically unstable problems in this book, *i.e.*, we only consider the numerical solution of dynamically stable problems.

Some Commonly Used FD Methods

We discuss the following finite difference methods for parabolic PDE in this chapter:

- the forward and backward Euler methods;
- the Crank–Nicolson and θ methods;
- the method of lines (MOL), provided a good ODE solver can be applied; and
- the alternating directional implicit (ADI) method, for high-dimensional problems.

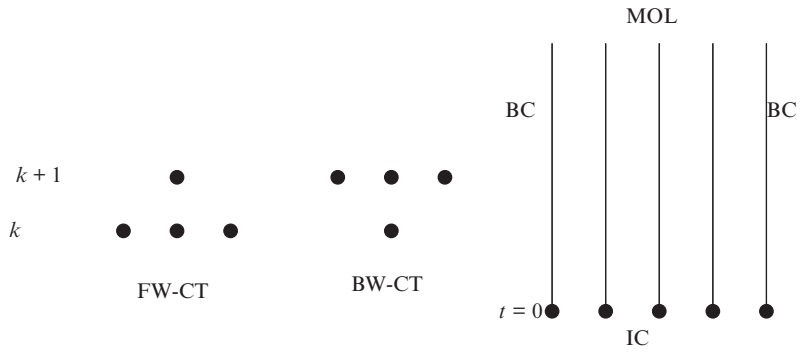

Fig. 4.1. Diagram of the finite difference stencil for the forward and backward Euler methods, and the MOL.

Finite difference methods applicable to elliptic PDEs can be used to treat the spatial discretization and boundary conditions, so let us focus on the time discretization and initial condition(s). To consider the stability of the consequent numerical methods, we invoke a Fourier transformation and von Neumann stability analysis.

4.1 The Euler Methods

For the following problem involving the heat equation with a source term,

$$u_t = \beta u_{xx} + f(x,t), \quad a < x < b, \quad t > 0,$$
$$u(a,t) = g_1(t), \quad u(b,t) = g_2(t), \qquad u(x,0) = u_0(x)\,,$$

let us seek a numerical solution for $u(x,t)$ at a particular time $T > 0$ or at certain times in the interval $0 < t < T$.

As the first step, we expect to generate a grid

$$x_i = a + ih, \quad i = 0, 1, \ldots, m, \quad h = \frac{b-a}{m},$$
$$t^k = k\Delta t, \quad k = 0, 1, \ldots, n\,, \quad \Delta t = \frac{T}{n}\,.$$

It turns out that we cannot use arbitrary Δt (even it may be small) for explicit methods because of numerical instability concerns. The second step is to approximate the derivatives with finite difference approximations. Since we already know how to discretize the spatial derivatives, let us focus on possible finite difference formulas for the time derivative. In Figure 4.1, we sketch the stencils of several finite difference methods.

4.1.1 Forward Euler Method (FW-CT)

At a grid point (x_i, t^k), $k > 0$, on using the forward finite difference approximation for u_t and central finite difference approximation for u_{xx} we have

$$\frac{u(x_i, t^k + \Delta t) - u(x_i, t^k)}{\Delta t} = \beta \frac{u(x_{i-1}, t^k) - 2u(x_i, t^k) + u(x_{i+1}, t^k)}{h^2} + f(x_i, t^k) + T(x_i, t^k)\,.$$

The local truncation error is

$$T(x_i, t^k) = -\frac{h^2\,\beta}{12} u_{xxxx}(x_i, t^k) + \frac{\Delta t}{2} u_{tt}(x_i, t^k) + \cdots,$$

where the dots denote higher-order terms, so the discretization is $O(h^2 + \Delta t)$. The discretization is first order in time and second order in space, when the finite difference equation is

$$\frac{U_i^{k+1} - U_i^k}{\Delta t} = \beta\, \frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{h^2} + f_i^k, \tag{4.3}$$

where $f_i^k = f(x_i, t^k)$, with U_i^k again denoting the approximate values for the true solution $u(x_i, t^k)$. When $k = 0$, U_i^0 is the initial condition at the grid point $(x_i, 0)$; and from the values U_i^k at the time level k the solution of the finite difference equation at the next time level $k + 1$ is

$$U_i^{k+1} = U_i^k + \Delta t \left(\beta \frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{h^2} + f_i^k \right), \quad i = 1, 2, \ldots, m-1\,. \tag{4.4}$$

The solution of the finite difference equations is thereby directly obtained from the approximate solution at previous time steps and we do not need to solve a system of algebraic equations, so the method is called *explicit*. Indeed, we successively compute the solution at t^1 from the initial condition at t^0, and then at t^2 using the approximate solution at t^1. Such an approach is often called a time marching method.

Remark 4.1. The local truncation error of the FW-CT finite difference scheme under our definition is

$$\begin{aligned} T(x,t) &= \frac{u(x, t + \Delta t) - u(x,t)}{\Delta t} - \beta\, \frac{u(x-h,t) - 2u(x,t) + u(x+h,t)}{h^2} - f(x,t) \\ &= O(h^2 + \Delta t)\,. \end{aligned}$$

In passing, we note an alternative definition of the truncation error in the literature

$$T(x,t) = u(x,t+\Delta t) - u(x,t) - \Delta t\left(\beta\frac{u(x-h,t)-2u(x,t)+u(x+h,t)}{h^2} - f(x,t)\right)$$
$$= O\left(\Delta t(h^2+\Delta t)\right)$$

introduces an additional factor Δt, so it is one order higher in Δt.

Remark 4.2. If $f(x,t)\equiv 0$ and β is a constant, then from $u_t = \beta u_{xx}$ and $u_{tt} = \beta\partial u_{xx}/\partial t = \beta\partial^2 u_t/\partial x^2 = \beta^2 u_{xxxx}$, the local truncation error is

$$T(x,t) = \left(\frac{\beta^2\Delta t}{2} - \frac{\beta h^2}{12}\right)u_{xxxx} + O\left((\Delta t)^2 + h^4\right). \tag{4.5}$$

Thus if β is constant we can choose $\Delta t = h^2/(6\beta)$ to get $O(h^4 + (\Delta t)^2) = O(h^4)$, *i.e.*, the local truncation error is fourth-order accurate without further computational complexity, which is significant for an *explicit* method.

It is easy to implement the forward Euler's method compared with other methods. Below we list some scripts of the Matlab file called *FW_Euler_heat.m*:

```
a = 0;  b=1; m = 10; n=20;
h = (b-a)/m;
k = h^2/2;  %k = h^2/1.9;

t = 0;  tau = k/h^2;
for i=1:m+1,
   x(i) = a + (i-1)*h;  y1(i) = uexact(t,x(i));  y2(i) = 0;
end
plot(x,y1); hold

for j=1:n,
   y1(1)=0; y1(m+1)=0;
   for i=2:m
     y2(i) = y1(i) + tau*(y1(i-1)-2*y1(i)+y1(i+1)) + k*f(t,x(i));
   end
   plot(x,y2); pause(0.25)
   t = t + k;  y1 = y2;
end
```

In the code above, we also plot the history of the solution. On testing the forward Euler method with different Δt and checking the error in a problem with a known exact solution, we find the method works well when $0 < \Delta t \le \frac{h^2}{2\beta}$ but blows up when $\Delta t > \frac{h^2}{2\beta}$. Since the method is consistent, we anticipate that

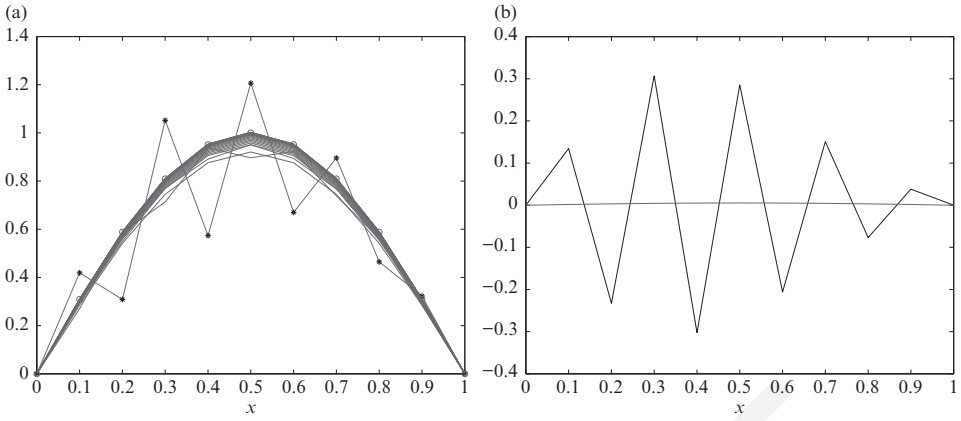

Fig. 4.2. (a) Plot of the computed solutions using two different time step sizes and the exact solution at some time for the test problem. (b) Error plots of the computed solution using the two different step sizes: one is stable and the error is small; the other one is unstable and the error grows rapidly.

this is a question of numerical stability. Intuitively, to prevent the errors in u_i^k being amplified, one can set

$$0 < \frac{2\beta\Delta t}{h^2} \le 1\,, \quad \text{or} \quad 0 < \Delta t \le \frac{h^2}{2\beta}\,. \tag{4.6}$$

This is a time step constraint, often called the CFL (Courant–Friedrichs–Lewy) stability condition, which can be verified numerically. In Figure 4.2, we plot the computed solution for a testing problem with $\beta = 1$, $f(x) = -\sin t \sin(\pi x) + \cos t \sin(\pi x)\pi^2$. The true solution is $u(x,t) = \cos t \sin(\pi x)$. We take 20 time marching steps using two different time steps, one is $\Delta t_1 = h^2/2$ (stable), and the other one is $\Delta t_2 = 2h^2$ (unstable). The left plots are the true solution at $t_1 = 20\Delta t_1$ and $t_2 = 20\Delta t_2$. The red lines are the history of the solution computed using $\Delta t_2 = 2h^2$, and the "*" indicates the computed solution at the grid points for the final step. We see that the solution begin to grow and oscillates. The plot of the blue line is the true solution at $t_1 = 20\Delta t_1$ with the little "o" as the finite difference solution at the grid points, which is first-order accurate. The right figure is the error plots with the blue one (the error is small) for the computed solution using $\Delta t_1 = h^2/2$; while the black one for the computed solution using $\Delta t_2 = 2h^2$, whose error grows rapidly and begin oscillates. If the final time T gets larger, so is the error, which we call the phenomenon as a blow-up due to the instability of the algorithm. The stability and the CFL condition of the time step constraint are very important for explicit or semi-explicit numerical algorithms.

4.1.2 The Backward Euler Method (BW-CT)

If the backward finite difference formula is used for u_t and the central finite difference approximation for u_{xx} at (x_i, t^k), we get

$$\frac{U_i^k - U_i^{k-1}}{\Delta t} = \beta \frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{h^2} + f_i^k, \quad k = 1, 2, \ldots,$$

which is conventionally reexpressed as

$$\frac{U_i^{k+1} - U_i^k}{\Delta t} = \beta \frac{U_{i-1}^{k+1} - 2U_i^{k+1} + U_{i+1}^{k+1}}{h^2} + f_i^{k+1}, \quad k = 0, 1, \ldots. \tag{4.7}$$

The backward Euler method is also consistent, and the discretization error is again $O(\Delta t + h^2)$.

Using the backward Euler method, we cannot get U_i^{k+1} with a few simple algebraic operations because all of the U_i^{k+1}'s are coupled together. Thus we need to solve the following tridiagonal system of equations, in order to get the approximate solution at the time level $k+1$:

$$\begin{bmatrix} 1+2\mu & -\mu & & & & \\ -\mu & 1+2\mu & -\mu & & & \\ & -\mu & 1+2\mu & -\mu & & \\ & \ddots & \ddots & \ddots & & \\ & & -\mu & 1+2\mu & -\mu & \\ & & & -\mu & 1+2\mu & \end{bmatrix} \begin{bmatrix} U_1^{k+1} \\ U_2^{k+1} \\ U_3^{k+1} \\ \vdots \\ U_{m-2}^{k+1} \\ U_{m-1}^{k+1} \end{bmatrix}$$

$$= \begin{bmatrix} U_1^k + \Delta t f_1^{k+1} + \mu g_1^{k+1} \\ U_2^k + \Delta t f_2^{k+1} \\ U_3^k + \Delta t f_3^{k+1} \\ \vdots \\ U_{m-2}^k + \Delta t f_{m-2}^{k+1} \\ U_{m-1}^k + \Delta t f_{m-1}^{k+1} + \mu g_2^{k+1} \end{bmatrix}, \tag{4.8}$$

where $\mu = \frac{\beta \Delta t}{h^2}$ and $f_i^{k+1} = f(x_i, t^{k+1})$. Note that we can use $f(x_i, t^k)$ instead of $f(x_i, t^{k+1})$, since the method is first-order accurate in time. Such a numerical method is called an *implicit*, because the solution at time level $k+1$ are coupled together. The advantage of the backward Euler method is that it is stable for any choice of Δt. For 1D problems, the computational cost is only slightly more than the explicit Euler method if we can use an efficient tridiagonal solver, such as the Grout factorization method at cost $O(5n)$ (*cf.* Burden and Faires, 2010, for example).

4.2 The Method of Lines

With a good solver for ODE or systems of ODEs, we can use the MOL to solve parabolic PDEs. In Matlab, we can use the ODE Suite to solve a system of ODEs. The ODE Suite contains Matlab functions such as ode23, ode23s, ode15s, ode45, and others. The Matlab function ode23 uses a combination of Runge–Kutta methods of order 2 and 3 and uses an adaptive time step size. The Matlab function ode23s is designed for a stiff system of ODE.

Consider a general parabolic equation of the form

$$u_t(x,t) = Lu(x,t) + f(x,t)\,,$$

where L is an elliptic operator. Let L_h be a corresponding finite difference operator acting on a grid $x_i = a + ih$. We can form a semidiscrete system of ODEs of form

$$\frac{\partial U_i}{\partial t} = L_h U_i(t) + f_i(t)\,,$$

where $U_i(t) \simeq u(x_i, t)$ is the spatial discretization of $u(x,t)$ along the line $x = x_i$, *i.e.*, we only discretize the spatial variable. For example, the heat equation with a source $u_t = \beta u_{xx} + f$ where $L = \beta \partial^2/\partial x^2$ is represented by $L_h = \beta \delta_{xx}^2$ produces the discretized system of ODE

$$\begin{aligned}
\frac{\partial U_1(t)}{\partial t} &= \beta \frac{-2U_1(t) + U_2(t)}{h^2} + \frac{g_1(t)}{h^2} + f(x_1, t)\,,\\
\frac{\partial U_i(t)}{\partial t} &= \beta \frac{U_{i-1}(t) - 2U_i(t) + U_{i+1}(t)}{h^2} + f(x_i, t), \quad i = 2, 3, \ldots, m-2\,,\\
\frac{\partial U_{m-1}(t)}{\partial t} &= \beta \frac{U_{m-2}(t) - 2U_{m-1}(t)}{h^2} + \frac{g_2(t)}{h^2} + f(x_{m-1}, t)\,,
\end{aligned} \tag{4.9}$$

and the initial condition is

$$U_i(0) = u_0(x_i, 0), \quad i = 1, 2, \ldots, m-1\,. \tag{4.10}$$

The ODE system can be written in the vector form

$$\frac{d\mathbf{y}}{dt} = f(\mathbf{y}, t), \qquad \mathbf{y}(0) = \mathbf{y}_0\,. \tag{4.11}$$

The MOL is especially useful for nonlinear PDEs of the form $u_t = f(\partial/\partial x, u, t)$. For linear problems, we typically have

$$\frac{d\mathbf{y}}{dt} = A\mathbf{y} + \mathbf{c}\,,$$

where A is a matrix and $\mathbf{c}$ is a vector. Both A and $\mathbf{c}$ may depend on t.

There are many efficient solvers for a system of ODEs. Most are based on high-order Runge–Kutta methods with adaptive time steps, *e.g.*, ODE suite in Matlab, or *dsode.f* available through *Netlib*. However, it is important to recognise that the ODE system obtained from the MOL is typically *stiff*, *i.e.*, the eigenvalues of A have very different scales. For example, for the heat equation the magnitude of the eigenvalues range from $O(1)$ to $O(1/h^2)$. In Matlab, we can call an ODE solver using the format

```
[t,y] = ode23s('yfun-mol', [0, t_final], y0);
```

The solution is stored in the last row of $\mathbf{y}$, which can be extracted using

```
[mr,nc] = size(y);
ysol = y(mr,:);
```

Then *ysol* is the approximate solution at time $t = t\_final$. To define the ODE system of the MOL, we can create a Matlab file, say yfun-mol.m whose contents contain the following

```
function yp = yfun-mol(t,y)
global m h x
k = length(y); yp=size(k,1);
yp(1) = (-2*y(1) + y(2))/(h*h) + f(t,x(1)) + g1(t)/(h*h);
for i=2:m-2
   yp(i) = (y(i-1) -2*y(1) + y(2))/(h*h) + f(t,x(i));
end
yp(m-1) = (y(m-2) -2*y(m-1) )/(h*h) + f(t,x(i)) + g2(t)/(h*h);
```

where $g1(t)$ and $g2(t)$ are two Matlab functions for the boundary conditions at $x = a$ and $x = b$; and $f(t, x)$ is the source term.

The initial condition can be defined as

```
global m h x
for i=1:m-1
    y0(i) = u_0(x(i));
end
```

where $u_0(x)$ is a Matlab function of the initial condition.

4.3 The Crank–Nicolson Scheme

The time step constraint $\Delta t = h^2/(2\beta)$ for the explicit Euler method is generally considered to be a severe restriction, *e.g.*, if $h = 0.01$, the final time is $T = 10$ and $\beta = 100$, then we need 2×10^7 steps to get the solution at the final time. The backward Euler method does not have the time step constraint, but it is only first-order accurate. If we want second-order accuracy $O(h^2)$, we need to take $\Delta t = O(h^2)$. One finite difference scheme that is second-order accurate both in space and time, without compromising stability and computational complexity, is the Crank–Nicolson scheme.

The Crank–Nicolson scheme is based on the following lemma, which can be proved easily using the Taylor expansion.

Lemma 4.3. *Let $\phi(t)$ be a function that has continuous first- and second-order derivatives, i.e., $\phi(t) \in C^2$. Then*

$$\phi(t) = \frac{1}{2}\left(\phi\left(t - \frac{\Delta t}{2}\right) + \phi\left(t + \frac{\Delta t}{2}\right)\right) + \frac{(\Delta t)^2}{8}u''(t) + h.o.t\,. \quad (4.12)$$

Intuitively, the Crank–Nicolson scheme approximates the PDE

$$u_t = (\beta u_x)_x + f(x,t)$$

at $(x_i, t^k + \Delta t/2)$, by averaging the time level t^k and t^{k+1} of the spatial derivative $\nabla \cdot (\beta \nabla u))$ and $f(x,t)$. Thus it has the following form

$$\frac{U_i^{k+1} - U_i^k}{\Delta t} = \frac{\beta^k_{i-\frac{1}{2}} U^k_{i-1} - (\beta^k_{i-\frac{1}{2}} + \beta^k_{i+\frac{1}{2}}) U^k_i + \beta^k_{i+\frac{1}{2}} U^k_{i+1}}{2h^2}$$

$$+ \frac{\beta^{k+1}_{i-\frac{1}{2}} U^{k+1}_{i-1} - (\beta^{k+1}_{i-\frac{1}{2}} + \beta^{k+1}_{i+\frac{1}{2}}) U^{k+1}_i + \beta^{k+1}_{i+\frac{1}{2}} U^{k+1}_{i+1}}{2h^2} + \frac{1}{2}\left(f_i^k + f_i^{k+1}\right). \quad (4.13)$$

The discretization is second order in time (central at $t + \Delta t/2$ with step size $\Delta t/2$) and second order in space. This can easily be seen using the following

relations, taking $\beta = 1$ for simplicity:

$$\frac{u(x,t+\Delta t)-u(x,t)}{\Delta t} = u_t(x,t+\Delta t/2) + \frac{1}{3}\left(\frac{\Delta t}{2}\right)^2 u_{ttt}(x,t+\Delta t/2) + O((\Delta t)^4),$$

$$\frac{u(x-h,t)-2u(x,t)+u(x+h,t)}{2h^2} = u_{xx}(x,t) + O(h^2),$$

$$\frac{u(x-h,t+\Delta t)-2u(x,t+\Delta t)+u(x+h,t+\Delta t)}{2h^2} = u_{xx}(x,t+\Delta t) + O(h^2)\,,$$

$$\frac{1}{2}\Big(u_{xx}(x,t)+u_{xx}(x,t+\Delta t)\Big) = u_{xx}(x,t+\Delta t/2) + O((\Delta t)^2),$$

$$\frac{1}{2}\Big(f(x,t)+f(x,t+\Delta t)\Big) = f(x,t+\Delta t/2) + O((\Delta t)^2)\,.$$

At each time step, we need to solve a tridiagonal system of equations to get U_i^{k+1}. The computational cost is only slightly more than that of the explicit Euler method in one space dimension, and we can take $\Delta t \simeq h$ and have second-order accuracy. Although the Crank–Nicolson scheme is an implicit method, it is much more efficient than the explicit Euler method since it is second-order accurate both in time and space with the same computational complexity. A sample Matlab code crank.m is accompanied with the book. If we use a fixed time step $\Delta t = h$, given a final time T, we can easily get the number of time marking steps as $N_T = int(T/h)$ as used in the crank.m. In the next section, we will prove it is unconditionally stable for the heat equation.

4.3.1 A Class of One-Step FD Methods: The θ-Method

The θ-method for the heat equation $u_t = u_{xx} + f(x,t)$ has the following form:

$$\frac{U_i^{k+1}-U_i^k}{\Delta t} = \theta\delta_{xx}^2 U_i^k + (1-\theta)\delta_{xx}^2 U_i^{k+1} + \theta f_i^k + (1-\theta) f_i^{k+1}\,.$$

When $\theta = 1$, the method is the explicit Euler method; when $\theta = 0$, the method is the backward Euler method; and when $\theta = 1/2$, it is the Crank–Nicolson scheme. If $0 < \theta \le 1/2$, then the method is unconditionally stable, and otherwise it is conditionally stable, *i.e.*, there is a time step constraint. The θ-method is generally first order in time and second order in space, except for $\theta = 1/2$.

The accompanying Matlab codes for this chapter included Euler, Crank–Nicolson, ADI, and MOL methods.

4.4 Stability Analysis for Time-Dependent Problems

A standard approach to stability analysis of finite difference methods for time-dependent problems is named after John von Neumann and based on the discrete Fourier transform (FT).

4.4.1 Review of the Fourier Transform

Let us first consider the Fourier transform in continuous space. Consider $u(x) \in L^2(-\infty, \infty)$, *i.e.*, $\int_{-\infty}^{\infty} u^2 dx < \infty$ or $\|u\|_2 < \infty$. The Fourier transform is defined as

$$\hat{u}(\omega) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{-i\omega x} u(x) dx \tag{4.14}$$

where $i = \sqrt{-1}$, mapping $u(x)$ in the space domain into $\hat{u}(\omega)$ in the frequency domain. Note that if a function is defined in the domain $(0, \infty)$ instead of $(-\infty, \infty)$, we can use the Laplace transform. The inverse Fourier transform is

$$u(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\omega x} \hat{u}(\omega) d\omega \,. \tag{4.15}$$

Parseval's relation: Under the Fourier transform, we have $\|\hat{u}\|_2 = \|u\|_2$ or

$$\int_{-\infty}^{\infty} |\hat{u}|^2 d\omega = \int_{-\infty}^{\infty} |u|^2 dx \,. \tag{4.16}$$

From the definition of the Fourier transform we have

$$\widehat{\left(\frac{\partial \hat{u}}{\partial \omega}\right)} = -ixu \,, \qquad \widehat{\frac{\partial u}{\partial x}} = i\omega \hat{u} \,. \tag{4.17}$$

To show this we invoke the inverse Fourier transform

$$\frac{\partial u(x)}{\partial x} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\omega x} \widehat{\frac{\partial u}{\partial x}} \, d\omega$$

so that, since $u(x)$ and $\hat{u}(\omega)$ are both in $L^2(-\infty, \infty)$, on taking the partial derivative of the inverse Fourier transform with respect to x we have

$$\frac{\partial u(x)}{\partial x} = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \frac{\partial}{\partial x} \left(e^{i\omega x} \widehat{u}\right) d\omega = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} i\omega \widehat{u} e^{i\omega x} \, d\omega \,.$$

Then as the Fourier transform and its inverse are unique, $\widehat{\partial u/\partial x} = i\omega \widehat{u}$. The proof of the first equality is left as an exercise. It is easy to generalize the

equality, to set

$$\widehat{\frac{\partial^m u}{\partial x^m}} = (i\omega)^m \hat{u} \tag{4.18}$$

i.e., we remove the derivatives of one variable.

The Fourier transform is a powerful tool to solve PDEs, as illustrated below.

Example 4.4. Consider

$$u_t + a u_x = 0\,, \qquad -\infty < x < \infty, \quad t > 0\,, \quad u(x,0) = u_0(x)$$

which is called an advection equation, or a one-way wave equation. This is a Cauchy problem since the spatial variable is defined in the entire space and $t \geq 0$. On applying the FT to the equation and the initial condition,

$$\widehat{u_t} + \widehat{a u_x} = 0, \quad \text{or} \quad \widehat{u}_t + a i\omega \hat{u} = 0, \qquad \hat{u}(\omega, 0) = \hat{u}_0(\omega)$$

i.e., we get an ODE

$$\hat{u}(\omega, t) = \hat{u}(\omega, 0)\, e^{-ia\omega t} = \hat{u}_0(\omega)\, e^{-ia\omega t}$$

for $\hat{u}(\omega)$. The solution to the original advection equation is thus

$$\begin{aligned} u(x,t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\omega x}\, \hat{u}_0(\omega)\, e^{-ia\omega t}\, d\omega \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\omega(x-at)}\, \hat{u}_0(\omega)\, d\omega \\ &= u(x - at, 0)\,, \end{aligned}$$

on taking the inverse Fourier transform. It is notable that the solution for the advection equation does not change shape, but simply propagates along the characteristic line $x - at = 0$, and that

$$\|u\|_2 = \|\hat{u}\|_2 = \|\hat{u}(\omega, 0) e^{-ia\omega t}\|_2 = \|\hat{u}(\omega, 0)\|_2 = \|u_0\|_2\,.$$

Example 4.5. Consider

$$u_t = \beta u_{xx}, \qquad -\infty < x < \infty, \quad t > 0, \quad u(x,0) = u_0(x), \quad \lim_{|x| \to \infty} u = 0\,,$$

involving the heat (or diffusion) equation. On again applying the Fourier transform to the PDE and the initial condition,

$$\widehat{u_t} = \widehat{\beta u_{xx}}, \quad \text{or} \quad \widehat{u}_t = \beta (i\omega)^2 \hat{u} = -\beta \omega^2 \hat{u}, \qquad \hat{u}(\omega, 0) = \hat{u}_0(\omega)\,,$$

and the solution of this ODE is

$$\hat{u}(\omega, t) = \hat{u}(\omega, 0)\, e^{-\beta \omega^2 t}\,.$$

Consequently, if $\beta > 0$, from the Parseval's relation, we have

$$\|u\|_2 = \|\hat{u}\|_2 = \|\hat{u}(\omega, 0)e^{-\beta\omega^2 t}\|_2 \leq \|u_0\|_2 .$$

Actually, it can be seen that $\lim_{t\to\infty} \|u\|_2 = 0$ and the second-order partial derivative term is call a diffusion or dissipative. If $\beta < 0$, then $\lim_{t\to\infty} \|u\|_2 = \infty$, the PDE is dynamically unstable.

Example 4.6. Dispersive waves.

Consider

$$u_t = \frac{\partial^{2m+1} u}{\partial x^{2m+1}} + \frac{\partial^{2m} u}{\partial x^{2m}} + l.o.t.,$$

where m is a nonnegative integer. For the simplest case $u_t = u_{xxx}$, we have

$$\widehat{u_t} = \widehat{\beta u_{xxx}}, \quad \text{or} \quad \widehat{u}_t = \beta(i\omega)^3 \hat{u} = -i\omega^3 \hat{u} ,$$

and the solution of this ODE is

$$\hat{u}(\omega, t) = \hat{u}(\omega, 0)\, e^{-i\omega^3 t} .$$

Therefore,

$$\|u\|_2 = \|\hat{u}\|_2 = \|\hat{u}(\omega, 0)\|_2 = \|u(\omega, 0)\|_2 ,$$

and the solution to the original PDE can be expressed as

$$\begin{aligned} u(x, t) &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\omega x}\, \hat{u}_0(\omega)\, e^{-i\omega^3 t}\, d\omega \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} e^{i\omega(x-\omega^2 t)}\, \hat{u}_0(\omega)\, d\omega . \end{aligned}$$

Evidently, the Fourier component with wave number ω propagates with velocity ω^2, so waves mutually interact but there is no diffusion.

Example 4.7. PDEs with higher-order derivatives.

Consider

$$u_t = \alpha \frac{\partial^{2m} u}{\partial x^{2m}} + \frac{\partial^{2m-1} u}{\partial x^{2m-1}} + l.o.t.,$$

where m is a nonnegative integer. The Fourier transform yields

$$\widehat{u_t} = \alpha(i\omega)^{2m}\hat{u} + \cdots = \begin{cases} -\alpha\omega^{2m}\hat{u} + \cdots & \text{if } m = 2k+1, \\ \alpha\omega^{2m}\hat{u} + \cdots & \text{if } m = 2k, \end{cases}$$

hence

$$\widehat{u} = \begin{cases} \hat{u}(\omega, 0)\, e^{-\alpha i \omega^{2m} t} + \cdots & \text{if } m = 2k+1, \\ \hat{u}(\omega, 0)\, e^{\alpha i \omega^{2m} t} + \cdots & \text{if } m = 2k, \end{cases}$$

such that $u_t = u_{xx}$ and $u_t = -u_{xxxx}$ are dynamically stable, whereas $u_t = -u_{xx}$ and $u_t = u_{xxxx}$ are dynamically unstable.

4.4.2 The Discrete Fourier Transform

Motivations to study a discrete Fourier transform include the stability analysis of finite difference schemes, data analysis in the frequency domain, filtering techniques, *etc.*

Definition 4.8. If $\ldots, v_{-2}, v_{-1}, v_0, v_1, v_2, \ldots$ denote the values of a continuous function $v(x)$ at $x_i = i\,h$, the discrete Fourier transform is defined as

$$\hat{v}(\xi) = \frac{1}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} h\, e^{-i\xi jh}\, v_j\,. \tag{4.19}$$

Remark 4.9.

- The definition is a quadrature approximation to the continuous case, i.e., we approximate $\int$ by $\sum$, and replace dx by h.
- $\hat{v}(\xi)$ is a continuous and periodic function of ξ with period $2\pi/h$, since

$$e^{-ijh(\xi + 2\pi/h)} = e^{-ijh\xi} e^{2ij\pi} = e^{-i\xi jh}\,, \tag{4.20}$$

so we can focus on $\hat{v}(\xi)$ in the interval $[-\pi/h, \pi/h]$, and consequently have the following definition.

Definition 4.10. The inverse discrete Fourier transform is

$$v_j = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi jh}\, \hat{v}(\xi)\, d\xi\,. \tag{4.21}$$

Given any finite sequence not involving h,

$$v_1, v_2, \ldots, v_M\,,$$

we can extend the finite sequence according to the following

$$\ldots, 0, 0, v_1, v_2, \ldots, v_M, 0, 0, \ldots\,,$$

and alternatively define the discrete Fourier and inverse Fourier transform as

$$\hat{v}(\xi) = \frac{1}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} e^{-i\xi j} v_j = \sum_{j=0}^{M} e^{-i\xi j}\, v_j\,, \tag{4.22}$$

$$v_j = \frac{1}{\sqrt{2\pi}} \int_{-\pi}^{\pi} e^{i\xi j}\, \hat{v}(\xi)\, d\xi\,. \tag{4.23}$$

We also define the *discrete norm* as

$$\|\mathbf{v}\|_h = \sqrt{\sum_{j=-\infty}^{\infty} v_j^2\, h}\,, \tag{4.24}$$

which is often denoted by $\|v\|_2$. Parseval's relation is also valid, *i.e.*,

$$\|\hat{v}\|_h^2 = \int_{-\pi/h}^{\pi/h} |\hat{v}(\xi)|^2 d\xi = \sum_{j=-\infty}^{\infty} h\,|v_j|^2 = \|v\|_h^2\,. \tag{4.25}$$

4.4.3 Definition of the Stability of a FD Scheme

A finite difference scheme $P_{\Delta t,h} v_j^k = 0$ is stable in a stability region Λ if for any positive time T there is an integer J and a constant C_T independent of Δt and h such that

$$\|\mathbf{v}^n\|_h \le C_T \sum_{j=0}^{J} \|\mathbf{v}^j\|_h\,, \tag{4.26}$$

for any n that satisfies $0 \le n\Delta t \le T$ with $(\Delta t, h) \in \Lambda$.

Remark 4.11.

1. The stability is usually independent of source terms.
2. A stable finite difference scheme means that the growth of the solution is at most a constant multiple of the sum of the norms of the solution at the first $J+1$ steps.
3. The stability region corresponds to all possible Δt and h for which the finite difference scheme is stable.

The following theorem provides a simple way to check the stability of any finite difference scheme.

Theorem 4.12. *If $\|\mathbf{v}^{k+1}\|_h \le \|\mathbf{v}^k\|_h$ is true for any k, then the finite difference scheme is stable.*

Proof: From the condition, we have

$$\|\mathbf{v}^n\|_h \le \|\mathbf{v}^{n-1}\|_h \le \cdots \le \|\mathbf{v}^1\|_h \le \|\mathbf{v}^0\|_h ,$$

and hence stability for $J = 0$ and $C_T = 1$.

4.4.4 The von Neumann Stability Analysis for FD Methods

The von Neumann stability analysis of a finite difference scheme can be sketched briefly as Discrete scheme $\Longrightarrow$ discrete Fourier transform $\Longrightarrow$ growth factor $g(\xi) \Longrightarrow$ stability ($|g(\xi)| \le 1$?). We will also explain a simplification of the von Neumann analysis.

Example 4.13. The forward Euler method (FW-CT) for the heat equation $u_t = \beta u_{xx}$ is

$$U_i^{k+1} = U_i^k + \mu \left(\frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{h^2} \right), \quad \mu = \frac{\beta \Delta t}{h^2} . \tag{4.27}$$

From the discrete Fourier transform, we have the following

$$U_j^k = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi jh} \hat{U}^k(\xi) d\xi , \tag{4.28}$$

$$U_{j+1}^k = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi (j+1)h} \hat{U}^k(\xi) d\xi = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi jh} e^{i\xi h} \hat{U}^k(\xi) d\xi , \tag{4.29}$$

and similarly

$$U_{j-1}^k = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi jh} e^{-i\xi h} \hat{U}^k(\xi) d\xi . \tag{4.30}$$

Substituting these relations into the forward Euler finite difference scheme, we obtain

$$U_i^{k+1} = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi jh} \left(1 + \mu(e^{-i\xi h} - 2 + e^{i\xi h}) \right) \hat{U}^k(\xi) d\xi . \tag{4.31}$$

On the other hand, from the definition of the discrete Fourier transform, we also know that

$$U_i^{k+1} = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} e^{i\xi jh} \hat{U}^{k+1}(\xi) d\xi . \tag{4.32}$$

The discrete Fourier transform is unique, which implies

$$\hat{U}^{k+1}(\xi) = \left(1 + \mu(e^{-i\xi h} - 2 + e^{i\xi h}) \right) \hat{U}^k(\xi) = g(\xi) \hat{U}^k(\xi) , \tag{4.33}$$

where

$$g(\xi) = 1 + \mu(e^{-i\xi h} - 2 + e^{i\xi h}) \tag{4.34}$$

is called the *growth factor*. If $|g(\xi)| \le 1$, then $|\hat{U}^{k+1}| \le |\hat{U}^k|$ and thus $\|\hat{\mathbf{U}}^{k+1}\|_h \le \|\hat{\mathbf{U}}^k\|_h$, so the finite difference scheme is stable.

Let us examine $|g(\xi)|$ now. We have

$$\begin{aligned} g(\xi) &= 1 + \mu\left(\cos(-\xi h) - i\sin(\xi h) - 2 + \cos(\xi h) + i\sin(\xi h)\right) \\ &= 1 + 2\mu\left(\cos(\xi h) - 1\right) = 1 - 4\mu\sin^2(\xi h)/2\,, \end{aligned} \tag{4.35}$$

but we need to know when $|g(\xi)| \le 1$, or $-1 \le g(\xi) \le 1$. Note that

$$-1 \le 1 - 4\mu \le 1 - 4\mu\sin^2(\xi h)/2 = g(\xi) \le 1\,, \tag{4.36}$$

so on taking $-1 \le 1 - 4\mu$ we can guarantee that $|g(\xi)| \le 1$, which implies the stability. Thus a sufficient condition for the stability of the forward Euler method is

$$-1 \le 1 - 4\mu \quad \text{or} \quad 4\mu \le 2, \quad \text{or} \quad \Delta t \le \frac{h^2}{2\beta}\,. \tag{4.37}$$

Although we cannot claim what will happen if this condition is violated, it provides an upper bound for the stability.

4.4.5 Simplification of the von Neumann Stability Analysis for One-Step Time Marching Methods

Consider the one-step time marching method $\mathbf{U}^{k+1} = f(\mathbf{U}^k, \mathbf{U}^{k+1})$. The following theorem provides a simple way to determine the stability.

Theorem 4.14. *Let $\theta = h\xi$. A one-step finite difference scheme (with constant coefficients) is stable if and only if there is a constant K (independent of θ, Δt, and h) and some positive grid spacing Δt_0 and h_0 such that*

$$|g(\theta, \Delta t, h)| \le 1 + K\Delta t \tag{4.38}$$

for all θ and $0 < h \le h_0$. If $g(\theta, \Delta t, h)$ is independent of h and Δt, then the stability condition (4.38) can be replaced by

$$|g(\theta| \le 1. \tag{4.39}$$

Thus only the amplification factor $g(h\xi) = g(\theta)$ needs to be considered, as observed by von Neumann.

The von Neumann stability analysis usually involves the following steps:

1. set $U_j^k = e^{ijh\xi}$ and substitute it into the finite difference scheme;
2. express U_j^{k+1} as $U_j^{k+1} = g(\xi)e^{ijh\xi}$, *etc.*;
3. solve for $g(\xi)$ and determine whether or when $|g(\xi)| \le 1$ (for stability); but note that
4. if there are some ξ such that $|g(\xi)| > 1$, then the method is unstable.

Example 4.15. The stability of the backward Euler method for the heat equation $u_t = \beta u_{xx}$ is

$$U_i^{k+1} = U_i^k + \mu\left(U_{i-1}^{k+1} - 2U_i^{k+1} + U_{i+1}^{k+1}\right), \quad \mu = \frac{\beta\Delta t}{h^2}. \tag{4.40}$$

Following the procedure mentioned above, we have

$$\begin{aligned} g(\xi)e^{ijh\xi} &= e^{ijh\xi} + \mu\left(e^{i\xi(j-1)h} - 2e^{i\xi jh} + e^{i\xi(j+1)h}\right) g(\xi) \\ &= e^{i\xi jh}\left(1 + \mu\left(e^{-i\xi h} - 2 + e^{i\xi h}i\right) g(\xi)\right), \end{aligned} \tag{4.41}$$

with solution

$$\begin{aligned} g(\xi) &= \frac{1}{1 - \mu(e^{-i\xi h} - 2 + e^{i\xi h})} \\ &= \frac{1}{1 - \mu(2\cos(h\xi) - 2)} = \frac{1}{1 + 4\mu\sin^2(h\xi)/2} \le 1, \end{aligned} \tag{4.42}$$

for any h and $\Delta t > 0$. Obviously, $-1 < 0 \le g(\xi)$ so $|g(\xi)| \le 1$ and the backward Euler method is unconditionally stable, *i.e.*, there is no constraint on Δt for stability.

Example 4.16. The Leapfrog scheme (two-stage method) for the heat equation $u_t = u_{xx}$ is

$$\frac{U_i^{k+1} - U_i^{k-1}}{2\Delta t} = \frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{h^2}, \tag{4.43}$$

involving the central finite difference formula both in time and space. This method is unconditionally unstable! To show this, we use $U_j^{k-1} = e^{ijh\xi}/g(\xi)$ to get

$$\begin{aligned} g(\xi)e^{ijh\xi} &= \frac{1}{g(\xi)}e^{ijh\xi} + e^{i\xi jh}\left(\mu(e^{-i\xi h} - 2 + e^{i\xi h})\right) \\ &= \frac{1}{g(\xi)}e^{ijh\xi} - e^{ijh\xi}4\mu\sin^2(h\xi/2), \end{aligned}$$

yielding a quadratic equation for $g(\xi)$:

$$(g(\xi))^2 + 4\mu \sin^2(h\xi/2)\, g(\xi) - 1 = 0. \tag{4.44}$$

The two roots are

$$g(\xi) = -2\mu \sin^2(h\xi/2) \pm \sqrt{4\mu^2 \sin^4(h\xi/2) + 1}\,,$$

and one root

$$g(\xi) = -2\mu \sin^2(h\xi/2) - \sqrt{4\mu^2 \sin^4(h\xi/2) + 1}$$

has magnitude $|g(\xi)| \geq 1$. Thus there are ξ such that $|g(\xi)| > 1$, so the method is unstable.

4.5 FD Methods and Analysis for 2D Parabolic Equations

The general form of a parabolic PDE is

$$u_t + a_1 u_x + a_2 u_y = (\beta u_x)_x + (\beta u_y)_y + \kappa u + f(x, y, t)\,,$$

with boundary conditions and an initial condition. We need $\beta \geq \beta_0 > 0$ for the dynamic stability. The PDE can be written as

$$u_t = Lu + f,$$

where L is the spatial differential operator. The MOL can be used provided there is a good solver for the stiff ODE system. Note that the system is large ($O(mn)$), if the numbers of grid lines are $O(m)$ and $O(n)$ in the x- and y-directions, respectively.

For simplicity, let us consider the heat equation $u_t = \nabla \cdot (\beta \nabla u) + f(x, y, t)$ and assume β is a constant. The simplest method is the forward Euler method:

$$U_{lj}^{k+1} = U_{lj}^k + \mu \left(U_{l-1,j}^k + U_{l+1,j}^k + U_{l,j-1}^k + U_{l,j+1}^k - 4U_{l,j}^k \right) + \Delta t f_{lj}^k\,,$$

where $\mu = \beta \Delta t / h^2$. The method is first order in time and second order in space, and it is conditionally stable. The stability condition is

$$\Delta t \leq \frac{h^2}{4\beta}. \tag{4.45}$$

Note that the factor is now 4, instead of 2 for 1D problems. To show stability using the von Neumann analysis with $f = 0$, set

$$u_{lj}^k = e^{i\,(lh_x\xi_1 + jh_y\xi_2)} = e^{i\,\xi\cdot\mathbf{x}} \tag{4.46}$$

where $\xi = [\xi_1,\ \xi_2]^T$ and $\mathbf{x} = [h_x l,\ h_y j]^T$,

$$U_{lj}^{k+1} = g(\xi_1, \xi_2)\, e^{i\,\xi\cdot\mathbf{x}}. \tag{4.47}$$

Note that the index is l instead of i in the x-direction, to avoid confusion with the imaginary unit $i = \sqrt{-1}$.

Substituting these expressions into the finite difference scheme, we obtain

$$g(\xi_1, \xi_2) = 1 - 4\mu \left(\sin^2(\xi_1 h/2) + \sin^2(\xi_2 h/2)\right),$$

where $h_x = h_y = h$ for simplicity. If we enforce

$$-1 \leq 1 - 8\mu \leq 1 - 4\mu \left(\sin^2(\xi_1 h/2) + \sin^2(\xi_2 h/2)\right) \leq 1 - 8\mu,$$

and take $-1 \leq 1 - 8\mu$, we can guarantee that $|g(\xi_1, \xi_2)| \leq 1$, which implies the stability. Thus, a sufficient condition for the stability of the forward Euler method in 2D is

$$\frac{8\Delta t \beta}{h^2} \leq 2, \quad \text{or} \quad \Delta t \leq \frac{h^2}{4\beta},$$

in addition to the condition $\Delta t > 0$.

4.5.1 The Backward Euler Method (BW-CT) in 2D

The backward Euler scheme can be written as

$$\frac{U_{ij}^{k+1} - U_{ij}^k}{\Delta t} = \frac{U_{i-1,j}^{k+1} + U_{i+1,j}^{k+1} + U_{i,j-1}^{k+1} + U_{i,j+1}^{k+1} - 4U_{ij}^{k+1}}{h^2} + f_{ij}^{k+1}, \tag{4.48}$$

which is first order in time and second order in space, and it is unconditionally stable. The coefficient matrix for the unknown U_{ij}^{k+1} is block tridiagonal, and strictly row diagonally dominant if the natural row ordering is used to index the U_{ij}^{k+1} and the finite difference equations.

4.5.2 The Crank–Nicolson (C–N) Scheme in 2D

The Crank–Nicolson scheme can be written as

$$\frac{U_{ij}^{k+1}-U_{ij}^{k}}{\Delta t}=\frac{1}{2}\left(\frac{U_{i-1,j}^{k+1}+U_{i+1,j}^{k+1}+U_{i,j-1}^{k+1}+U_{i,j+1}^{k+1}-4U_{ij}^{k+1}}{h^2}+f_{ij}^{k+1}\right.$$
$$\left.+\frac{U_{i-1,j}^{k}+U_{i+1,j}^{k}+U_{i,j-1}^{k}+U_{i,j+1}^{k}-4U_{ij}^{k}}{h^2}+f_{ij}^{k}\right). \quad (4.49)$$

Both the local truncation error and global error are $\mathrm{O}\big((\Delta t)^2+h^2\big)$. The scheme is unconditionally stable for linear problems. However, we need to solve a system of equations with a strictly row diagonally dominant and block tridiagonal coefficient matrix, if we use the natural row ordering for both the equations and unknowns.

A structured multigrid method can be applied to solve the linear system of equations from the backward Euler method or the Crank–Nicolson scheme.

4.6 The ADI Method

The ADI is a *time splitting* or *fractional step* method. The idea is to use an implicit discretization in one direction and an explicit discretization in another direction. For the heat equation $u_t=u_{xx}+u_{yy}+f(x,y,t)$, the ADI method is

$$\frac{U_{ij}^{k+\frac{1}{2}}-U_{ij}^{k}}{(\Delta t)/2}=\frac{U_{i-1,j}^{k+\frac{1}{2}}-2U_{ij}^{k+\frac{1}{2}}+U_{i+1,j}^{k+\frac{1}{2}}}{h_x^2}+\frac{U_{i,j-1}^{k}-2U_{ij}^{k}+U_{i,j+1}^{k}}{h_y^2}+f_{ij}^{k+\frac{1}{2}},$$
$$\frac{U_{ij}^{k+1}-U_{ij}^{k+\frac{1}{2}}}{(\Delta t)/2}=\frac{U_{i-1,j}^{k+\frac{1}{2}}-2U_{ij}^{k+\frac{1}{2}}+U_{i+1,j}^{k+\frac{1}{2}}}{h_x^2}+\frac{U_{i,j-1}^{k+1}-2U_{ij}^{k+1}+U_{i,j+1}^{k+1}}{h_y^2}+f_{ij}^{k+\frac{1}{2}}, \quad (4.50)$$

which is second order in time and in space if $u(x,y,t)\in C^4(\Omega)$, where Ω is the bounded domain where the PDE is defined. It is unconditionally stable for linear problems. We can use symbolic expressions to discuss the method,

rewritten as

$$\begin{aligned} U_{ij}^{k+\frac{1}{2}} &= U_{ij}^{k} + \frac{\Delta t}{2}\delta_{xx}^2 U_{ij}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\delta_{yy}^2 U_{ij}^{k} + \frac{\Delta t}{2} f_{ij}^{k+\frac{1}{2}}, \\ U_{ij}^{k+1} &= U_{ij}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\delta_{xx}^2 U_{ij}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\delta_{yy}^2 U_{ij}^{k+1} + \frac{\Delta t}{2} f_{ij}^{k+\frac{1}{2}}. \end{aligned} \tag{4.51}$$

Thus on moving unknowns to the left-hand side, in matrix-vector form we have

$$\begin{aligned} \left(I - \frac{\Delta t}{2}D_x^2\right)\mathbf{U}^{k+\frac{1}{2}} &= \left(I + \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k} + \frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}}, \\ \left(I - \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k+1} &= \left(I + \frac{\Delta t}{2}D_x^2\right)\mathbf{U}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}}, \end{aligned} \tag{4.52}$$

leading to a simple analytically convenient result as follows. From the first equation we get

$$\mathbf{U}^{k+\frac{1}{2}} = \left(I - \frac{\Delta t}{2}D_x^2\right)^{-1}\left(I + \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k} + \left(I - \frac{\Delta t}{2}D_x^2\right)^{-1}\frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}},$$

and substituting into the second equation to have

$$\begin{aligned} \left(I - \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k+1} = \left(I + \frac{\Delta t}{2}D_x^2\right)\left(I - \frac{\Delta t}{2}D_x^2\right)^{-1}\left(I + \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k} \\ + \left(I + \frac{\Delta t}{2}D_x^2\right)\left(I - \frac{\Delta t}{2}D_x^2\right)^{-1}\frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}}. \end{aligned}$$

We can go further to get

$$\begin{aligned} \left(I - \frac{\Delta t}{2}D_x^2\right)\left(I - \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k+1} = \left(I + \frac{\Delta t}{2}D_x^2\right)\left(I + \frac{\Delta t}{2}D_y^2\right)\mathbf{U}^{k} \\ + \left(I + \frac{\Delta t}{2}D_x^2\right)\frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\mathbf{F}^{k+\frac{1}{2}}. \end{aligned}$$

This is the equivalent one step time marching form of the ADI method, which will be use to show the stability of the ADI method later. Note that in this derivation we have used

$$\left(I + \frac{\Delta t}{2}D_x^2\right)\left(I + \frac{\Delta t}{2}D_y^2\right) = \left(I + \frac{\Delta t}{2}D_y^2\right)\left(I + \frac{\Delta t}{2}D_x^2\right)$$

and other commutative operations.

4.6.1 Implementation of the ADI Algorithm

The key idea of the ADI method is to use the implicit discretization dimension by dimension by taking advantage of fast tridiagonal solvers. In the x-direction, the finite difference approximation is

$$U_{ij}^{k+\frac{1}{2}} = U_{ij}^{k} + \frac{\Delta t}{2}\delta_{xx}^2 U_{ij}^{k+\frac{1}{2}} + \frac{\Delta t}{2}\delta_{yy}^2 U_{ij}^{k} + \frac{\Delta t}{2} f_{ij}^{k+\frac{1}{2}}.$$

For a fixed j, we get a tridiagonal system of equations for $U_{1j}^{k+\frac{1}{2}}$, $U_{2j}^{k+\frac{1}{2}}$, …, $U_{m-1,j}^{k+\frac{1}{2}}$, assuming a Dirichlet boundary condition at $x=a$ and $x=b$. The system of equations in matrix-vector form is

$$\begin{bmatrix} 1+2\mu & -\mu & & & & & \\ -\mu & 1+2\mu & -\mu & & & & \\ & -\mu & 1+2\mu & -\mu & & & \\ & & \ddots & \ddots & \ddots & & \\ & & & -\mu & 1+2\mu & -\mu & \\ & & & & -\mu & 1+2\mu & \end{bmatrix} \begin{bmatrix} U_{1j}^{k+\frac{1}{2}} \\ U_{2j}^{k+\frac{1}{2}} \\ U_{3j}^{k+\frac{1}{2}} \\ \vdots \\ U_{m-2,j}^{k+\frac{1}{2}} \\ U_{m-1,j}^{k+\frac{1}{2}} \end{bmatrix} = \widehat{\mathbf{F}},$$

where

$$\widehat{\mathbf{F}} = \begin{bmatrix} U_{1,j}^{k} + \frac{\Delta t}{2} f_{1j}^{k+\frac{1}{2}} + \mu\, u_{bc}(a, y_j)^{k+\frac{1}{2}} + \mu\left(U_{1,j-1}^{k} - 2U_{1,j}^{k} + U_{1,j+1}^{k}\right) \\ U_{2,j}^{k} + \frac{\Delta t}{2} f_{2j}^{k+\frac{1}{2}} + \mu\left(U_{2,j-1}^{k} - 2U_{2,j}^{k} + U_{2,j+1}^{k}\right) \\ U_{3j}^{k} + \frac{\Delta t}{2} f_{3j}^{k+\frac{1}{2}} + \mu\left(U_{3,j-1}^{k} - 2U_{3,j}^{k} + U_{3,j+1}^{k}\right) \\ \vdots \\ U_{m-2,j}^{k} + \frac{\Delta t}{2} f_{m-2,j}^{k+\frac{1}{2}} + \mu\left(U_{m-2,j-1}^{k} - 2U_{m-2,j}^{k} + U_{m-2,j+1}^{k}\right) \\ U_{m-1,j}^{k} + \frac{\Delta t}{2} f_{m-1,j}^{k+\frac{1}{2}} + \mu\left(U_{m-1,j-1}^{k} - 2U_{m-1,j}^{k} + U_{m-1,j+1}^{k}\right) + \mu\, u_{bc}(b, y_j)^{k+\frac{1}{2}} \end{bmatrix},$$

and $\mu = \frac{\beta \Delta t}{2h^2}$, and $f_i^{k+\frac{1}{2}} = f(x_i, t^{k+\frac{1}{2}})$. For each j, we need to solve a symmetric tridiagonal system of equations. The cost for the x-sweep is about $O(5mn)$.

4.6.1.1 A Pseudo-code of the ADI Method in Matlab

```
for j = 2:n,                                % Loop for fixed j
   A = sparse(m-1,m-1); b=zeros(m-1,1);
   for i=2:m,
      b(i-1) = (u1(i,j-1) -2*u1(i,j) + u1(i,j+1))/h1 + ...
                f(t2,x(i),y(j)) + 2*u1(i,j)/dt;
      if i == 2
        b(i-1) = b(i-1) + uexact(t2,x(i-1),y(j))/h1;
        A(i-1,i) = -1/h1;
      else
        if i==m
          b(i-1) = b(i-1) + uexact(t2,x(i+1),y(j))/h1;
          A(i-1,i-2) = -1/h1;
        else
           A(i-1,i) = -1/h1;
           A(i-1,i-2) = -1/h1;
        end
      end
      A(i-1,i-1) = 2/dt + 2/h1;
    end
    ut = A\b;                               % Solve the diagonal matrix.

%-------------- loop in the y direction --------------------------
for i = 2:m,
   A = sparse(m-1,m-1); b=zeros(m-1,1);
   for j=2:n,
      b(j-1) = (u2(i-1,j) -2*u2(i,j) + u2(i+1,j))/h1 + ...
                f(t2,x(i),y(j)) + 2*u2(i,j)/dt;
      if j == 2
        b(j-1) = b(j-1) + uexact(t1,x(i),y(j-1))/h1;
        A(j-1,j) = -1/h1;
      else
        if j==n
          b(j-1) = b(j-1) + uexact(t1,x(i),y(j+1))/h1;
          A(j-1,j-2) =  -1/h1;
        else
           A(j-1,j) = -1/h1;
           A(j-1,j-2) = -1/h1;
        end
      end
      A(j-1,j-1) = 2/dt + 2/h1;             % Solve the system
    end
    ut = A\b;
```

A Matlab test code adi.m can be found in the depository directory of this chapter.

4.6.2 Consistency of the ADI Method

Adding the two equations in (4.50) together, we get

$$\frac{U_{ij}^{k+1} - U_{ij}^{k}}{(\Delta t)/2} = 2\delta_{xx}^2 U_{ij}^{k+\frac{1}{2}} + \delta_{yy}^2 \left(U_{ij}^{k+1} + U_{ij}^{k} \right) + 2f_{ij}^{k+\frac{1}{2}}; \tag{4.53}$$

and if we subtract the first equation from the second equation, we get

$$4U_{ij}^{k+\frac{1}{2}} = 2\left(U_{ij}^{k+1} + U_{ij}^{k} \right) - \Delta t \delta_{yy}^2 \left(U_{ij}^{k+1} - U_{ij}^{k} \right). \tag{4.54}$$

Substituting this into (4.53) we get

$$\left(1 + \frac{(\Delta t)^2}{4} \delta_{xx}^2 \delta_{yy}^2 \right) \frac{U_{ij}^{k+1} - U_{ij}^{k}}{\Delta t} = \left(\delta_{xx}^2 + \delta_{yy}^2 \right) \frac{U_{ij}^{k+1} - U_{ij}^{k}}{2} + f_{ij}^{k+\frac{1}{2}}, \tag{4.55}$$

and we can clearly see that the discretization is second-order accurate in both space and time, *i.e.*, $T_{ij}^{k} = O((\Delta t)^2 + h^2)$.

4.6.3 Stability Analysis of the ADI Method

Taking $f = 0$ and setting

$$U_{lj}^{k} = e^{i(\xi_1 h_1 l + \xi_2 h_2 j)}, \qquad U_{lj}^{k+1} = g(\xi_1, \xi_2)\, e^{i(\xi_1 h_1 l + \xi_2 h_2 j)}, \tag{4.56}$$

on using the operator form we have

$$\left(1 - \frac{\Delta t}{2} \delta_{xx}^2 \right) \left(1 - \frac{\Delta t}{2} \delta_{yy}^2 \right) \mathbf{U}_{jl}^{k+1} = \left(1 + \frac{\Delta t}{2} \delta_{xx}^2 \right) \left(1 + \frac{\Delta t}{2} \delta_{yy}^2 \right) \mathbf{U}_{jl}^{k},$$

which yields,

$$\begin{aligned} &\left(1 - \frac{\Delta t}{2} \delta_{xx}^2 \right) \left(1 - \frac{\Delta t}{2} \delta_{yy}^2 \right) g(\xi_1, \xi_2)\, e^{i(\xi_1 h_1 l + \xi_2 h_2 j)} \\ &\qquad = \left(1 + \frac{\Delta t}{2} \delta_{xx}^2 \right) \left(1 + \frac{\Delta t}{2} \delta_{yy}^2 \right) e^{i(\xi_1 h_1 l + \xi_2 h_2 j)}. \end{aligned}$$

After some manipulations, we get

$$g(\xi_1, \xi_2) = \frac{\left(1 - 4\mu \sin^2(\xi_1 h/2) \right) \left(1 - 4\mu \sin^2(\xi_2 h/2) \right)}{\left(1 + 4\mu \sin^2(\xi_1 h/2) \right) \left(1 + 4\mu \sin^2(\xi_2 h/2) \right)},$$

where $\mu = \frac{\Delta t}{2h^2}$ and for simplicity we have set $h_x = h_y = h$. Thus $|g(\xi_1, \xi_2)| \leq 1$, no matter what Δt and h are, so the ADI method is unconditionally stable for linear heat equations.

4.7 An Implicit–Explicit Method for Diffusion and Advection Equations

Consider a diffusion and advection PDE in 2D

$$u_t + \mathbf{w} \cdot \nabla u = \nabla \cdot (\beta \nabla u) + f(x, y, t)$$

where $\mathbf{w}$ is a 2D vector, and ∇ is the gradient operator in 2D, see page 48. In this case, it is not so easy to get a second-order implicit scheme such that the coefficient matrix is diagonally dominant or symmetric positive or negative definite, due to the advection term $\mathbf{w} \cdot \nabla u$. One approach is to use an implicit scheme for the diffusion term and an explicit scheme for the advection term, of the following form from time level t^k to t^{k+1}:

$$\frac{u^{k+1} - u^k}{\Delta t} + (\mathbf{w} \cdot \nabla_h u)^{k+\frac{1}{2}} = \frac{1}{2} \left((\nabla_h \cdot \beta \nabla_h u)^k + (\nabla_h \cdot \beta \nabla_h u)^{k+1} \right) + f^{k+\frac{1}{2}}, \tag{4.57}$$

where

$$(\mathbf{w} \cdot \nabla_h u)^{k+\frac{1}{2}} = \frac{3}{2} (\mathbf{w} \cdot \nabla_h u)^k - \frac{1}{2} (\mathbf{w} \cdot \nabla_h u)^{k-1}, \tag{4.58}$$

where $\nabla_h u = [\delta_x u, \delta_y u]^T$, and at a grid point (x_i, y_j), they are

$$\delta_x u = \frac{u_{i+1,j} - u_{i-1,j}}{2h_x}; \qquad \delta_x u = \frac{u_{i,j+1} - u_{i,j-1}}{2h_y}. \tag{4.59}$$

We treat the advection term implicitly, since the term only contains the first-order partial derivatives and the CFL constraint is not a main concern unless $\|\mathbf{w}\|$ is very large. The time step constraint is

$$\Delta t \leq \frac{h}{2\|\mathbf{w}\|_2}. \tag{4.60}$$

At each time step, we need to solve a generalized Helmholtz equation

$$(\nabla \cdot \beta \nabla u)^{k+1} - \frac{2u^{k+1}}{\Delta t} = -\frac{2u^k}{\Delta t} + 2 (\mathbf{w} \cdot \nabla u)^{k+\frac{1}{2}} - (\nabla \cdot \beta \nabla u)^k - 2f^{k+\frac{1}{2}}. \tag{4.61}$$

We need $\mathbf{u}^1$ to get the scheme above started. We can use the explicit Euler method (FW-CT) to approximate $\mathbf{u}^1$, as this should not affect the stability and global error $O((\Delta t)^2 + h^2)$.

4.8 Solving Elliptic PDEs using Numerical Methods for Parabolic PDEs

We recall the steady-state solution of a parabolic PDE is the solution of the corresponding elliptic PDE, *e.g.*, the steady-state solution of the parabolic PDE

$$u_t = \nabla \cdot (\beta \nabla u) + \mathbf{w} \cdot \nabla u + f(\mathbf{x}, t)$$

is the solution to the elliptic PDE

$$\nabla \cdot (\beta \nabla u) + \mathbf{w} \cdot \nabla u + \bar{f}(\mathbf{x}) = 0\,,$$

if the limit

$$\bar{f}(\mathbf{x}) = \lim_{t \to \infty} f(\mathbf{x}, t)$$

exists. The initial condition is irrelevant to the steady-state solution, but the boundary condition is relevant. This approach has some advantages, especially for nonlinear problems where the solution is not unique. We can control the variation of the intermediate solutions, and the linear system of equations is more diagonally dominant. Since we only require the steady-state solution, we prefer to use implicit methods with large time steps since the accuracy in time is unimportant.

Exercises

1. Show that a scheme for

$$u_t = \beta\, u_{xx} \tag{4.62}$$

of the form

$$U_i^{k+1} = \alpha U_i^k + \frac{1-\alpha}{2}\left(U_{i+1}^k + U_{i-1}^k\right)$$

where $\alpha = 1 - 2\beta\mu,\ \mu = \Delta t/h^2$ is consistent with the heat equation (4.62). Find the order of the discretization.

2. Show that the implicit scheme

$$\left(1 - \frac{\Delta t \beta}{2}\,\delta_{xx}^2\right)\left(\frac{U_i^{k+1} - U_i^k}{\Delta t}\right) = \beta\left(1 - \frac{h^2}{12}\,\delta_{xx}^2\right)\delta_{xx}^2 U_i^k \tag{4.63}$$

for the heat equation (4.62) has order of accuracy $((\Delta t)^2, h^4)$, where

$$\delta_{xx}^2 U_i = \frac{U_{i-1} - 2U_i + U_{i+1}}{h^2},$$

and $\Delta t = O(h^2)$. Compare this method with FW-CT, BW-CT, and Crank–Nicolson schemes and explain the advantages and limitations. (**Note:** The stability condition of the scheme is $\beta\mu \leq \frac{3}{2}$).

3. For the implicit Euler method applied to the heat equation $u_t = u_{xx}$, is it possible to choose Δt such that the discretization is $O((\Delta t)^2 + h^4)$?
4. Consider the diffusion and advection equation

$$u_t + u_x = \beta u_{xx}, \quad \beta > 0. \tag{4.64}$$

Use the von Neumann analysis to derive the time step restriction for the scheme

$$\frac{U_i^{k+1} - U_i^k}{\Delta t} + \frac{U_{i+1}^k - U_{i-1}^k}{2h} = \beta \frac{U_{i-1}^k - 2U_i^k + U_{i+1}^k}{h^2}.$$

5. Implement and compare the ***Crank–Nicolson*** and the ***MOL*** methods using Matlab for the heat equation:

$$u_t = \beta u_{xx} + f(x,t), \quad a < x < b, \quad t \geq 0,$$
$$u(x,0) = u_0(x); \quad u(a,t) = g_1(t); \quad u_x(b,t) = g_2(t),$$

where β is a constant. Use $u(x,t) = (\cos t)\, x^2 \sin(\pi x)$, $0 < x < 1$, $tfinal = 1.0$ to test and debug your code. Write a short report about these two methods. Your discussion should include the grid refinement analysis, error and solution plots for $m = 80$, comparison of cputime, and any conclusions you can draw from your results. You can use Matlab code *ode15s* or *ode23s* to solve the semidiscrete ODE system.
Assume that u is the temperature of a thin rod with one end ($x = b$) just heated. The other end of the rod has a room temperature ($70^\circ C$). Solve the problem and find the history of the solution. Roughly how long does it take for the temperature of the rod to reach the steady state? What is the exact solution of the steady state? **Hint:** Take the initial condition as $u(x,0) = T_0\, e^{-(x-b)^2/\gamma}$, where T_0 and γ are two constants, $f(x,t) = 0$, and the Neumann boundary condition $u_x(b,t) = 0$.
6. Carry out the von Neumann analysis to determine the stability of the θ method

$$\frac{U_j^{(n+1)} - U_j^n}{k} = b\left(\theta\, \delta_x^2 U_j^{(n)} + (1-\theta)\, \delta_x^2 U_j^{(n+1)}\right) \tag{4.65}$$

for the heat equation $u_t = b u_{xx}$, where

$$\delta_x^2 U_j = \frac{U_{j-1} - 2U_j + U_{j+1}}{h^2} \quad \text{and} \quad 0 \leq \theta \leq 1.$$

7. Modify the Crank–Nicolson Matlab code for the backward Euler's method and for variable $\beta(x,t)$'s in one space dimensions. Validate your code.
8. Implement and compare the ADI and Crank–Nicolson methods with the SOR(ω) (try to test optimal ω) for the following problem involving the 2D heat equation:

$$u_t = u_{xx} + u_{yy} + f(t,x,y), \quad a < x < b, \quad c \leq y \leq d, \quad t \geq 0,$$
$$u(0,x,y) = u_0(x,y),$$

and Dirichlet boundary conditions. Choose two examples with known exact solutions to test and debug your code. Write a short report about the two methods. Your discussion should include the grid refinement analysis (with a fixed final time, say $T = 0.5$), error and solution plots, comparison of cpu time and flops, and any conclusions you can draw from your results.

9. **Extra credit:** Modify the ADI Matlab code for variable heat conductivity $\beta(x, y)$.