part

4A

Chapter 12

Diffusion Equations

We now begin to study finite difference methods for time-dependent partial differential equations, where variations in space are related to variations in time. We begin with the heat equation (or diffusion equation) introduced in Chapter 10,

$$u_t = \kappa u_{xx}. \tag{12.1}$$

This is the classical example of a *parabolic* equation, and many of the general properties seen here carry over to the design of numerical methods for other parabolic equations. We will assume $\kappa = 1$ for simplicity but some comments will be made about how the results scale to other values of $\kappa > 0$. (If $\kappa < 0$ then (12.1) would be a "backward heat equation", which is an ill-posed problem.)

Along with this equation we need initial conditions at some time t_0, which we typically take to be $t_0 = 0$,

$$u(x,0) = \eta(x) \tag{12.2}$$

and also boundary conditions if we are working on a bounded domain, *e.g.*, the Dirichlet conditions

$$\begin{aligned} u(0,t) &= g_0(t) \quad \text{for } t > 0 \\ u(1,t) &= g_1(t) \quad \text{for } t > 0 \end{aligned} \tag{12.3}$$

if $0 \le x \le 1$.

We have already studied the steady state version of this equation and spatial discretizations of u_{xx} (Chapter 2). We have also studied discretizations of the time derivatives and some of the stability issues that arise with these discretizations in Chapters 6 through 9. Next we will put these two types of discretizations together.

In practice we generally apply a set of finite difference equations on a discrete grid with grid points (x_i, t_n) where

$$x_i = ih, \qquad t_n = nk.$$

Here $h = \Delta x$ is the mesh spacing on the x-axis and $k = \Delta t$ is the time step. Let $U_i^n \approx u(x_i, t_n)$ represent the numerical approximation at grid point (x_i, t_n).

Since the heat equation is an evolution equation that can be solved forward in time, we set up our difference equations in a form where we can march forward in time, determining the values U_i^{n+1} for all i from the values U_i^n at the previous time level, or perhaps using also values at earlier time levels with a multistep formula.

As an example, one natural discretization of (12.1) would be

$$\frac{U_i^{n+1} - U_i^n}{k} = \frac{1}{h^2}(U_{i-1}^n - 2U_i^n + U_{i+1}^n). \tag{12.4}$$

This uses our standard centered difference in space and a forward difference in time. This is an *explicit* method since we can compute each U_i^{n+1} explicitly in terms of the previous data:

$$U_i^{n+1} = U_i^n + \frac{k}{h^2}(U_{i-1}^n - 2U_i^n + U_{i+1}^n). \tag{12.5}$$

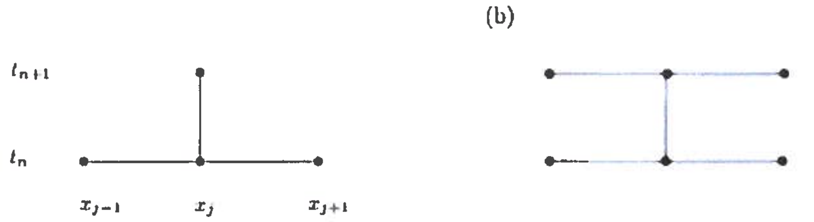

Figure 12.1: Stencils for the methods (12.5) and (12.7).

Figure 12.1(a) shows the *stencil* of this method. This is a one-step method in time, which is also called a two-level method in the context of PDE's since it involves the solution at two different time levels.

Another one-step method, which is much more useful in practice as we will see below, is the *Crank-Nicolson* method,

$$\begin{aligned}\frac{U_i^{n+1}-U_i^n}{k} &= \frac{1}{2}(D^2U_i^n + D^2U_i^{n+1}) \qquad (12.6)\\ &= \frac{1}{2h^2}(U_{i-1}^n - 2U_i^n + U_{i+1}^n + U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}),\end{aligned}$$

which can be rewritten as

$$U_i^{n+1} = U_i^n + \frac{k}{2h^2}(U_{i-1}^n - 2U_i^n + U_{i+1}^n + U_{i-1}^{n+1} - 2U_i^{n+1} + U_{i+1}^{n+1}) \qquad (12.7)$$

or

$$-rU_{i-1}^{n+1} + (1+2r)U_i^{n+1} - rU_{i+1}^{n+1} = rU_{i-1}^n + (1-2r)U_i^n + rU_{i+1}^n \qquad (12.8)$$

where $r = k/2h^2$. This is an *implicit* method and gives a tridiagonal system of equations to solve for all the values U_i^{n+1} simultaneously. In matrix form this is

$$\begin{bmatrix} (1+2r) & -r & & & & \\ -r & (1+2r) & -r & & & \\ & -r & (1+2r) & -r & & \\ & & \ddots & \ddots & \ddots & \\ & & & -r & (1+2r) & -r \\ & & & & -r & (1+2r) \end{bmatrix} \begin{bmatrix} U_1^{n+1} \\ U_2^{n+1} \\ U_3^{n+1} \\ \vdots \\ U_{m-1}^{n+1} \\ U_m^{n+1} \end{bmatrix}$$
$$= \begin{bmatrix} r(g_0(t_n)+g_0(t_{n+1})) + (1-2r)U_1^n + rU_2^n \\ rU_1^n + (1-2r)U_2^n + rU_3^n \\ rU_2^n + (1-2r)U_3^n + rU_4^n \\ \vdots \\ rU_{m-2}^n + (1-2r)U_{m-1}^n + rU_m^n \\ rU_{m-1}^n + (1-2r)U_m^n + r(g_1(t_n)+g_1(t_{n+1})) \end{bmatrix}. \qquad (12.9)$$

Note how the boundary conditions $u(0,t) = g_0(t)$ and $u(1,t) = g_1(t)$ come into these equations.

Since a tridiagonal system of m equations can be solved with $O(m)$ work, this method is essentially as efficient per time step as an explicit method. We will see in Section 12.4 that the heat equation is "stiff", and hence this implicit method, which allows much larger time steps to be taken than an explicit method, is a very efficient method for the heat equation.

Solving a parabolic equation with an implicit method requires solving a system of equations with the same structure as the 2-point boundary value problem we studied in Chapter 2. Similarly, a multidimensional parabolic equation requires solving a problem with the structure of a multidimensional elliptic equation in each time step. See Section 12.7.

Solving a parabolic equation with an implicit method requires solving a system of equations with the same structure as the 2-point boundary value problem we studied in Chapter 2. Similarly, a multidimensional parabolic equation requires solving a problem with the structure of a multidimensional elliptic equation in each time step.

12.1 Local truncation errors and order of accuracy

We can define the local truncation error as usual — we insert the exact solution $u(x,t)$ of the PDE into the finite difference equation and determine by how much it fails to satisfy the discrete equation.

Example 12.1. The local truncation error of the method (12.5) is based on the form (12.4): $\tau_i^n = \tau(x_i, t_n)$, where

$$\tau(x,t) = \frac{u(x,t+k)-u(x,t)}{k} - \frac{1}{h^2}(u(x-h,t) - 2u(x,t) + u(x+h,t)).$$

Again we should be careful to use the form that directly models the differential equation in order to get powers of k and h that agree with what we hope to see in the global error. Although we don't know $u(x,t)$ in general, if we assume it is smooth and use Taylor series expansions about $u(x,t)$, we find that

$$\tau(x,t) = \left(u_t + \frac{1}{2}ku_{tt} + \frac{1}{6}k^2u_{ttt} + \cdots\right) - \left(u_{xx} + \frac{1}{12}h^2u_{xxxx} + \cdots\right).$$

Since $u_t = u_{xx}$, the $O(1)$ terms drop out. By differentiating $u_t = u_{xx}$ we find that $u_{tt} = u_{txx} = u_{xxxx}$ and so

$$\tau(x,t) = \left(\frac{1}{2}k - \frac{1}{12}h^2\right)u_{xxxx} + O(k^2 + h^4).$$

This method is said to be *second order accurate in space* and *first order accurate in time* since the truncation error is $O(h^2 + k)$.

The Crank-Nicolson method is centered in both space and time, and an analysis of its local truncation error (Exercise 12.1) shows that it is second order accurate in both space and time,

$$\tau(x,t) = O(k^2 + h^2).$$

A method is said to be *consistent* if $\tau(x,t) \to 0$ as $k,\ h \to 0$. Just as in the other cases we have studied (boundary value problems and initial value problems for ODE's), we expect that consistency, plus some form of stability, will be enough to prove that the method converges at each fixed point (X,T) as we refine the grid in both space and time. Moreover we expect that for a stable method the global order of accuracy will agree with the order of the local truncation error, *e.g.*, for Crank-Nicolson we expect that

$$U_i^n - u(X,T) = O(k^2 + h^2)$$

as $k,\ h \to 0$ when $ih \equiv X$ and $nk \equiv T$ are fixed.

For linear PDE's, the fact that consistency plus stability is equivalent to convergence is known as the *Lax Equivalence Theorem*, and is discussed in Section 12.5 after introducing the proper concept of stability. As usual, it is the definition and study of stability that is the hard (and interesting) part of this theory.

12.2 Method of Lines discretizations

To understand how stability theory for time-dependent PDE's relates to the stability theory we have already developed for time-dependent ODE's, it is easiest to first consider the so-called Method of Lines (MOL) discretization of the PDE. In this approach we first discretize in space alone, which gives a large

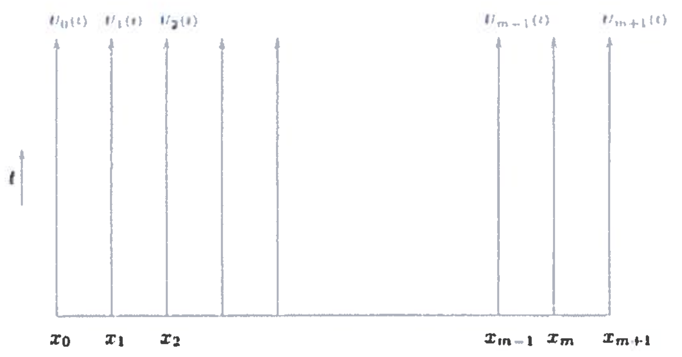

Figure 12.2: Method of lines interpretation. $U_i(t)$ is the solution along the line forward in time at the grid point x_i.

system of ODE's with each component of the system corresponding to the solution at some grid point, as a function of time. The system of ODE's can then be solved using one of the methods for ODE's that we have previously studied.

For example, we might discretize the heat equation (12.1) in space at grid point x_i by

$$U_i'(t) = \frac{1}{h^2}(U_{i-1}(t) - 2U_i(t) + U_{i+1}(t)), \quad \text{for } i = 1, 2, \ldots, m, \tag{12.10}$$

where prime now means differentiation with respect to time. We can view this as a coupled system of m ODE's for the variables $U_i(t)$, which vary continuously in time along the lines shown in Figure 12.2. This system can be written as

$$U'(t) = AU(t) + g(t) \tag{12.11}$$

where the tridiagonal matrix A is exactly as in (2.9) and $g(t)$ includes the terms needed for the boundary conditions, $U_0(t) \equiv g_0(t)$ and $U_{m+1}(t) \equiv g_1(t)$,

$$A = \frac{1}{h^2}\begin{bmatrix} -2 & 1 & & & & \\ 1 & -2 & 1 & & & \\ & 1 & -2 & 1 & & \\ & & \ddots & \ddots & \ddots & \\ & & & 1 & -2 & 1 \\ & & & & 1 & -2 \end{bmatrix}, \quad g(t) = \frac{1}{h^2}\begin{bmatrix} g_0(t) \\ 0 \\ 0 \\ \vdots \\ 0 \\ g_1(t) \end{bmatrix}. \tag{12.12}$$

This MOL approach is sometimes used in practice by first discretizing in space and then applying a software package for systems of ODE's. There are also packages that are specially designed to apply MOL. This approach has the advantage of being relatively easy to apply to a fairly general set of time-dependent PDE's, but the resulting method is often not as efficient as specially designed methods for the PDE.

As a tool in understanding stability theory, however, the MOL discretization is extremely valuable, and this is the main use we will make of it. We know how to analyze the stability of ODE methods applied to a linear system of the form (12.11) based on the eigenvalues of the matrix A, which now depend on the spatial discretization.

If we apply an ODE method to discretize the system (12.11), we will obtain a fully discrete method which produces approximations $U_i^n \approx U_i(t_n)$ at discrete points in time which are exactly the points (x_i, t_n) of the grid that we introduced at the beginning of this chapter.

For example, applying Euler's method $U^{n+1} = U^n + kf(U^n)$ to this linear system results in the fully discrete method (12.5). Applying instead the trapezoidal method (6.16) results in the Crank-Nicolson method (12.7). Applying a higher order linear multistep or Runge-Kutta method would give a different method, though with the spatial discretization (12.10) the overall method would be only second order accurate in space. Replacing the right hand side of (12.10) by a higher-order approximation to $u_{xx}(x_i)$ and then using a higher order time discretization would give a more accurate method.

12.3 Stability theory

We can now investigate the stability of schemes like (12.5) or (12.7) since these can be interpreted as standard ODE methods applied to the linear system (12.11). We expect the method to be stable if $k\lambda \in S$, *i.e.*, if the time step k multiplied by any eigenvalue λ of A lies in the stability region of the ODE method, as discussed in Chapter 8.

We have determined the eigenvalues of A in (2.23),

$$\lambda_p = \frac{2}{h^2}(\cos(p\pi h) - 1), \quad \text{for } p = 1,\, 2,\, \ldots,\, m, \tag{12.13}$$

where again m and h are related by $h = 1/(m+1)$. Note that there is a new wrinkle here relative to the ODE's we considered in Chapter 8: the eigenvalues λ_p depend on the mesh width h. As we refine the grid and $h \to 0$, the dimension of A increases, the number of eigenvalues we must consider increases, and the values of the eigenvalues change.

This is something we must bear in mind when we attempt to prove convergence as $k,\ h \to 0$. To begin with, however, let's consider the simpler question of how the method behaves for some fixed k and h, *i.e.*, the question of absolute stability in the ODE sense. Then it is clear that the method is absolutely stable (*i.e.*, the effect of past errors will not grow exponentially in future time steps) provided that $k\lambda_p \in S$ for each p, where S is the stability region of the ODE method, as discussed in Chapter 8.

For the matrix (12.12) coming from the heat equation, the eigenvalues lie on the negative real axis and the one farthest from the origin is $\lambda_m \approx -4/h^2$. Hence we require that $-4k/h^2 \in S$ (assuming the stability region is connected along the negative real axis up to the origin, as is generally the case).

Example 12.2. If we use Euler's method to obtain the discretization (12.5), then we must require $|1 + k\lambda| \le 1$ for each eigenvalue (see Chapter 8) and hence we require $-2 \le -4k/h^2 \le 0$. This limits the time step allowed to

$$\frac{k}{h^2} \le \frac{1}{2}. \tag{12.14}$$

This is a severe restriction: the time step must decrease like h^2 as we refine the grid, which is much smaller than the spatial width h when h is small.

Example 12.3. If we use the trapezoidal method we obtain the Crank-Nicolson discretization (12.6). The trapezoidal method for the ODE is absolutely stable in the whole left half plane and the eigenvalues (12.13) are always negative. Hence the Crank-Nicolson method is stable for *any* time step $k > 0$. Of course it may not be accurate if k is too large. Generally we must take $k = O(h)$ to obtain a reasonable solution, and the unconditional stability allows this.

12.4 Stiffness of the heat equation

Note that the system of ODE's we are solving is quite stiff, particularly for small h. The eigenvalues of A lie on the negative real axis with one fairly close to the origin, $\lambda_1 \approx -\pi^2$ for all h, while the largest in magnitude is $\lambda_m \approx -4/h^2$. The "stiffness ratio" of the system is $4/\pi^2 h^2$, which grows rapidly as $h \to 0$. As a result the explicit Euler method is stable only for very small time steps $k \le \frac{1}{2}h^2$. This is typically much smaller than what we would like to use over physically meaningful times, and an implicit method designed for stiff problems will be more efficient.

The stiffness is a reflection of the very different time scales present in solutions to the physical problem modelled by the heat equation. High frequency spatial oscillations in the initial data will decay very rapidly due to rapid diffusion over very short distances, while smooth data decays much more slowly since diffusion over long distances takes much longer. This is apparent from the Fourier analysis of Section 11.3 or is easily seen by writing down the exact solution to the heat equation on $0 \le x \le 1$ with $g_0(t) = g_1(t) \equiv 0$ as a Fourier sine series:

$$u(x,t) = \sum_{j=1}^{\infty} \hat{u}_j(t)\sin(j\pi x).$$

Inserting this in the heat equation gives the ODE's

$$\hat{u}_j'(t) = -j^2\pi^2\hat{u}_j(t), \qquad \text{for } j = 1,\ 2,\ \ldots \tag{12.15}$$

and so

$$\hat{u}_j(t) = e^{-j^2\pi^2 t}\hat{u}_j(0),$$

with the $\hat{u}_j(0)$ determined as the Fourier coefficients of the initial data $\eta(x)$.

We can view the equations (12.15) as an infinite system of ODE's, but which are decoupled so that the coefficient matrix is diagonal, with eigenvalues $-j^2\pi^2$ for $j = 1,\ 2,\ \ldots$. By choosing data with sufficiently rapid oscillation (large j), we can obtain arbitrarily rapid decay. For general initial data there may be some transient period when any high wave numbers are rapidly damped, but then the long-time behavior is dominated by the slower decay rates. See Figure 12.3 for some examples of the time evolution with different sets of data.

If we are solving the problem over the long time periods needed to track this slow diffusion, then we would ultimately (after any physical transients have decayed) like to use rather large time steps, since typically the variation in time is then on roughly the same scale as variations in space. We would generally like to have $k \approx h$ so that we have roughly the same resolution in time as we do in space. A method that requires $k \approx h^2$ forces us to take a much finer temporal discretization that we should need to represent smooth solutions. If $h = 0.001$, for example, then if we must take $k = h^2$ rather than $k = h$ we would need to take 1000 time steps to cover each time interval that should be well modelled by a single time step. This is the same difficulty we encountered with stiff ODE's in Chapter 9.

Note: The remark above that we want $k \approx h$ is reasonable assuming the method we are using has the same order of accuracy in both space and time. The method (12.5) does not have this property. Since the error is $O(k + h^2)$ we might want to take $k = O(h^2)$ just to get the same level of accuracy in both space and time. In this sense the stability restriction $k = O(h^2)$ may not seem unreasonable, but this is simply another reason for not wanting to use this particular method in practice.

Note: The general diffusion equation is $u_t = \kappa u_{xx}$ and in practice the diffusion coefficient κ may be different from 1 by many orders of magnitude. How does this affect our conclusions above? We would expect by scaling considerations that we should take $k \approx h/\kappa$ in order to achieve comparable resolution in space and time, *i.e.*, we would like to take $\kappa k/h \approx 1$. (Note that $\hat{u}_j(t) = \exp(-j^2\pi^2\kappa t)\hat{u}_j(0)$ in this case.) With the MOL discretization we obtain the system (12.11) but A now has a factor κ in front. For stability we thus require $-4\kappa k/h^2 \in S$, which requires $\kappa k/h^2$ to be order 1 for any explicit method. This is smaller than what we wish to use by a factor of h, regardless of the magnitude of κ. So our conclusions on stiffness are unchanged by κ. In particular, even when the diffusion coefficient is very small it is best to use an implicit method because we then want to take very long time steps $k \approx h/\kappa$.

These comments apply to the case of pure diffusion. If we are solving an advection-diffusion or reaction-diffusion equation where there are other time scales determined by other phenomena, then if the diffusive term has a very small coefficient we may be able to use an explicit method efficiently because of other restrictions on the time step.

Note: The physical problem of diffusion is "infinitely stiff" in the sense that there are eigenvalues $-j^2\pi^2$ with arbitrarily large magnitude since j can be any integer. Luckily the discrete problem is not this stiff. The reason it is not is that, once we discretize in space, only a finite number of spatial wave numbers can be represented and we obtaine the finite set of eigenvalues (12.13). As we refine the grid we can represent higher and higher wave numbers, leading to the increasing stiffness ratio as $h \to 0$.

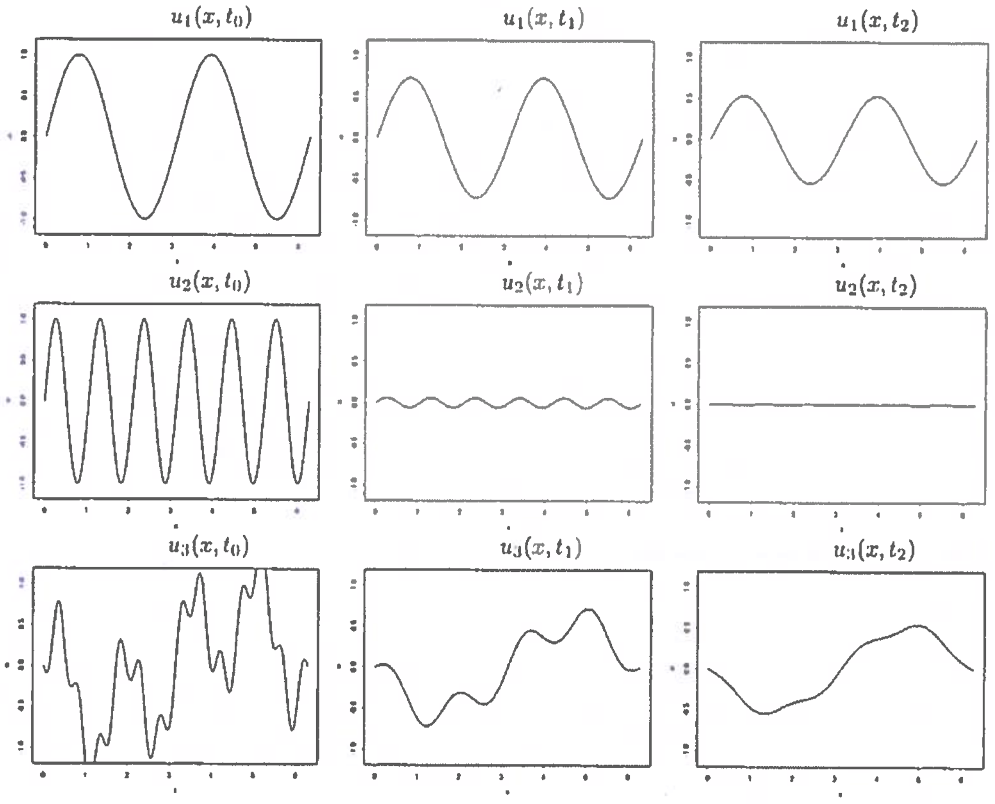

Figure 12.3: Solutions to the heat equation at three different times (columns) shown for three different sets of initial conditions (rows). In the top row $u_1(x,t_0)$ consists of only a low wave number, which decays slowly. The middle row shows data consisting of a higher wave number, which decays more quickly. The bottom row shows data $u_3(x,t_0)$ that contains a mixture of wave numbers. The high wave numbers are most rapidly damped (an initial rapid transient) while at later times only the lower wave numbers are still visible and decaying slowly.

12.5 Convergence

So far we have only discussed absolute stability, and determined the relation between k and h that must be satisfied to ensure that errors do not grow exponentially as we march forward in time on this fixed grid. We now address the question of convergence at a fixed point (X, T) as the grid is refined. It turns out that in general exactly the same relation between k and h must now be required to hold as we vary k and h, letting both go to zero.

In other words, we cannot let k and h go to zero at arbitrary independent rates and necessarily expect the resulting approximations to converge to the solution of the PDE. For a particular sequence of grids (k_1, h_1), (k_2, h_2), $\ldots$, with $k_j \to 0$ and $h_j \to 0$, we will expect convergence only if the proper relation ultimately holds for each pair. For the method (12.5), for example, the sequence of approximations will converge only if $k_j/h_j^2 \leq 1/2$ for all j sufficiently large.

It is sometimes easiest to think of k and h as being related by some fixed rule (*e.g.*, we might choose $k = 0.4h^2$ for the method (12.5)), so that we can speak of convergence as $k \to 0$ with the understanding that this relation holds on each grid.

The methods we have studied so far can be written in the form

$$U^{n+1} = BU^n + b^n \tag{12.16}$$

for some matrix $B \in \mathbb{R}^{m\times m}$ on a grid with $h = 1/(m+1)$ and $b^n \in \mathbb{R}^m$. In the usual way, we can apply the difference equation to the exact solution $u(x,t)$ and obtain

$$u^{n+1} = Bu^n + b^n + k\tau^n \tag{12.17}$$

where

$$u^n = \begin{bmatrix} u(x_1,t_n) \\ u(x_2,t_n) \\ \vdots \\ u(x_m,t_n) \end{bmatrix}, \qquad \tau^n = \begin{bmatrix} \tau(x_1,t_n) \\ \tau(x_2,t_n) \\ \vdots \\ \tau(x_m,t_n) \end{bmatrix}.$$

Subtracting (12.17) from (12.16) gives the difference equation for the global error $E^n = U^n - u^n$:

$$E^{n+1} = BE^n - k\tau^n,$$

and hence, as usual,

$$E^n = B^n E^0 - k\sum_{m=1}^{n} B^{n-m}\tau^{m-1},$$

from which we obtain

$$\|E^n\| \leq \|B^n\|\|E^0\| + k\sum_{m=1}^{n} \|B^{n-m}\|\,\|\tau^{m-1}\|. \tag{12.18}$$

The method *converges* provided it is *consistent*, which requires that $\tau^{n-1} \to 0$ in each step, and *stable*, which now requires that $\|B^n\|$ be uniformly bounded for all k and n with $nk \leq T$. In the context of linear PDE's, the fact that consistency together with this form of stability gives convergence is known as the *Lax Equivalence Theorem*. A complete proof can be found in [RM67], but the essential inequality is (12.18). The form of stability required here, the uniform bound on $\|B^n\|$, is often called *Lax-Richtmyer stability* in the present context.

Recall that B depends on both k and h, but we are assuming some fixed relationship between these. For the methods we analyzed earlier in this Chapter, we found relations that would guarantee $\|B\|_2 \leq 1$ for each pair k, h, from which Lax-Richtmyer stability follows directly (in the 2-norm at least).

12.5.1 PDE vs. ODE stability theory

It may bother you that the stability we need for convergence now seems to depend on absolute stability, and on the shape of the stability region for the time-discretization, which determines the required relationship between k and h. Recall that in the case of ODE's all we needed for convergence was "zero-stability", which does not depend on the shape of the stability region except for the requirement that the point $z = 0$ must lie in this region.

Here is the difference: With ODE's we were studying a fixed system of ODE's and dimension of the system and the fixed set of eigenvalues λ were independent of k. For convergence we needed $k\lambda$ in the stability region as $k \to 0$, but since these values all converge to 0 it is only the origin that is important, at least in order to prove convergence as $k \to 0$. Hence the need for zero-stabilty. With PDE's, on the other hand, in our MOL discretization the system of ODE's grows as we refine the grid, and the eigenvalues λ grow as k and h go to zero. So it is not clear that $k\lambda$ will go to zero, and zero-stability is not sufficient. For the heat equation with k/h^2 fixed, these values do not go to zero as $k \to 0$. For convergence we must now require that these values at least lie in the region of absolute stability as $k \to 0$, and this gives the stability restriction relating k and h. If we want to keep k/h fixed as $k, h \to 0$, then $k\lambda \to -\infty$ and we must use an implicit method, and one that includes the entire negative real axis in its stability region.

Although for the methods considered so far we have obtained $\|B\| \leq 1$, this is not really necessary in order to have Lax-Richtmyer stability. If there is a constant α so that a bound of the form

$$\|B\| \leq 1 + \alpha k \tag{12.19}$$

holds in some norm (at least for all k sufficiently small), then we will have Lax-Richtmyer stability in this norm, since

$$\|B^n\| \leq (1 + \alpha k)^n \leq e^{\alpha T}$$

for $nk \leq T$. Since the matrix B depends on k and grows in size as $k \to 0$, the general theory of stability in the sense of uniform power boundedness of such families of matrices is often nontrivial. The *Kreiss Matrix Theorem* is one important tool in many practical problems. This is discussed in [RM67] along with some other techniques. See also [Str89] for a good discussion of stability. The recent review paper [LT98] gives an overview of how ODE and PDE stability theory are related, with a discussion of stability theory based on the "energy method", another important approach.

12.6 von Neumann analysis

Although it is useful to go through the MOL formulation in order to understand how stability theory for PDE's is related to the theory for ODE's, in practice there is another approach that will typically give the proper stability restrictions more easily.

The von Neumann approach to stability analysis is based on Fourier analysis and hence is generally limited to constant coefficient linear PDE's. For simplicity it is usually applied to the *Cauchy problem*, which is the PDE on all space with no boundaries, $-\infty < x < \infty$ in the one-dimensional case. Von Neumann analysis can also be used to study the stability of problems with *periodic boundary conditions*, e.g., in $0 \leq x \leq 1$ with $u(0,t) = u(1,t)$ imposed. This is generally equivalent to a Cauchy problem with periodic initial data.

Stability theory for PDE's with more general boundary conditions can often be quite difficult, as the coupling between the discretization of the boundary conditions and the discretization of the PDE can be very subtle. Von Neumann analysis addresses the issue of stability of the PDE discretization alone. Some discussion of stability theory for initial boundary value problems can be found in [Str89], [RM67].

The Cauchy problem for linear PDE's can be solved using Fourier transforms — see Chapter 11 for a review. The basic reason this works is that the functions $e^{i\xi x}$ with wave number ξ = constant are eigenfunctions of the differential operator ∂_x,

$$\partial_x e^{i\xi x} = i\xi e^{i\xi x},$$

and hence of any constant coefficient linear differential operator. Von Neumann analysis is based on the fact that the related grid function $W_j = e^{ijh\xi}$ is an eigenfunction of any standard finite difference operator[1]. For example, if we approximate $v'(x_j)$ by $D_0 V_j = \frac{1}{2h}(V_{j+1} - V_{j-1})$, then in general the grid function $D_0 V$ is not just a scalar multiple of V. But for the special case of W, we obtain

$$\begin{aligned} D_0 W_j &= \frac{1}{2h}\left(e^{i(j+1)h\xi} - e^{i(j-1)h\xi}\right) \\ &= \frac{1}{2h}\left(e^{ih\xi} - e^{-ih\xi}\right) e^{ijh\xi} \\ &= \frac{i}{h}\sin(h\xi) e^{ijh\xi} \\ &= \frac{i}{h}\sin(h\xi) W_j. \end{aligned} \tag{12.20}$$

So W is an "eigengridfunction" of the operator D_0, with eigenvalue $\frac{i}{h}\sin(h\xi)$.

Note the relation between these and the eigenfunctions and eigenvalues of the operator ∂_x found earlier: W_j is simply the eigenfunction $w(x)$ of ∂_x evaluated at the point x_j, and for small $h\xi$ we can approximate the eigenvalue of D_0 by

$$\begin{aligned} \frac{i}{h}\sin(h\xi) &= \frac{i}{h}\left(h\xi - \frac{1}{6}h^3\xi^3 + O(h^5\xi^5)\right) \\ &= i\xi - \frac{i}{6}h^2\xi^3 + \cdots . \end{aligned}$$

This agrees with the eigenvalue $i\xi$ of ∂_x to $O(h^2\xi^3)$.

Suppose we have a grid function V_j defined at grid points $x_j = jh$ for $j = 0, \pm 1, \pm 2, \ldots$, which is an l_2 function in the sense that the 2-norm

$$\|U\|_2 = \left(h \sum_{j=-\infty}^{\infty} |U_j|^2\right)^{1/2}$$

is finite. Then we can express V_j as a linear combination of the grid functions $e^{ijh\xi}$ for all ξ in the range $-\pi/h \le \xi \le \pi/h$. Functions with larger wave number ξ cannot be resolved on this grid. We can write

$$V_j = \frac{1}{\sqrt{2\pi}} \int_{-\pi/h}^{\pi/h} \hat{V}(\xi) e^{ijh\xi}\, d\xi$$

where

$$\hat{V}(\xi) = \frac{h}{\sqrt{2\pi}} \sum_{j=-\infty}^{\infty} V_j e^{-ijh\xi}.$$

These are direct analogs of the formulas for a function $v(x)$ in the discrete case.

Again we have *Parseval's relation*, $\|\hat{V}\|_2 = \|V\|_2$, although the 2-norms used for the grid function V_j and the function $\hat{V}(\xi)$ defined on $[-\pi/h, \pi/h]$ are different:

$$\|V\|_2 = \left(h \sum_{j=-\infty}^{\infty} |V_j|^2\right)^{1/2}, \qquad \|\hat{V}\|_2 = \left(\int_{-\pi/h}^{\pi/h} |\hat{V}(\xi)|^2\, d\xi\right)^{1/2}.$$

[1]Note: in this section $i = \sqrt{-1}$ and the index j is used on the grid functions.

In order to show that a finite difference method is stable in the 2-norm by the techniques discussed earlier in this chapter, we would have to show that $\|B\|_2 \leq 1 + \alpha k$ in the notation of (12.19). This amounts to showing that there is a constant α such that

$$\|U^{n+1}\|_2 \leq (1+\alpha k)\|U^n\|_2$$

for all U^n. This can be difficult to attack directly because of the fact that computing $\|U\|_2$ requires summing over all grid points, and each U_j^{n+1} depends on values of U^n at neighboring grid points so that all grid points are coupled together. In some cases one can work with these infinite sums directly, but it is rare that this can be done. Alternatively one can work with the matrix B itself, as we did above in Section 12.5, but this matrix is growing as we refine the grid.

Using Parseval's relation, we see that it is sufficient to instead show that

$$\|\hat{U}^{n+1}\|_2 \leq (1+\alpha k)\|\hat{U}^n\|_2$$

where $\hat{U}^n$ is the Fourier transform of the grid function U^n. The utility of Fourier analysis now stems from the fact that after Fourier transforming the finite difference method, we obtain a recurrence relation for each $\hat{U}^n(\xi)$ that is decoupled from all other wave numbers. For a 2-level method this has the form

$$\hat{U}^{n+1}(\xi) = g(\xi)\hat{U}^n(\xi). \tag{12.21}$$

The factor $g(\xi)$, which depends on the method, is called the *amplification factor* for the method at wave number ξ. If we can show that

$$|g(\xi)| \leq 1 + \alpha k$$

where α is independent of ξ, then it follows that the method is stable, since then

$$|\hat{U}^{n+1}(\xi)| \leq (1+\alpha k)|\hat{U}^n(\xi)| \quad \text{for all } \xi$$

and so

$$\|\hat{U}^{n+1}\|_2 \leq (1+\alpha k)\|\hat{U}^n\|_2.$$

Fourier analysis allows us to obtain simple scalar recursions of the form (12.21) for each wave number separately, rather than dealing with a system of equations for U_j^n that couples together all values of j.

Note: Here we are assuming that $u(x,t)$ is a scalar, so that $g(\xi)$ is a scalar. For an system of s equations we would find that $g(\xi)$ is an $s \times s$ matrix for each value of ξ, so some analysis of matrix eigenvalues is still required to investigate stability. But the dimension of the matrices is s, independent of the grid spacing, unlike the MOL analysis where the matrix dimension increases as $h \to 0$.

Example 12.4. Consider the method (12.5). To apply von Neumann analysis we consider how this method works on a single wavenumber ξ, *i.e.*, we set

$$U_j^n = e^{ijh\xi}. \tag{12.22}$$

Then we expect that

$$U_j^{n+1} = g(\xi)e^{ijh\xi}, \tag{12.23}$$

where $g(\xi)$ is the amplification factor for this wavenumber. Inserting these expressions into (12.5) gives

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

and hence

$$g(\xi) = 1 + 2\frac{k}{h^2}(\cos(\xi h) - 1).$$

Since $-1 \le \cos(\xi h) \le 1$ for any value of ξ, we see that

$$1 - 4\frac{k}{h^2} \le g(\xi) \le 1$$

for all ξ. We can guarantee that $|g(\xi)| \le 1$ for all ξ if we require

$$4\frac{k}{h^2} \le 2.$$

This is exactly the stability restriction (12.14) we found earlier for this method. If this restriction is violated, then the Fourier components with some wave number ξ will be amplified (and, as expected, it is the largest wavenumbers that go unstable first as k is increased).

Example 12.5. The fact that the Crank-Nicolson method is stable for all k and h can also be shown using von Neumann analysis. Substituting (12.22) and (12.23) into the difference equations (12.7) and cancelling the common factor of $e^{ijh\xi}$ gives the following relation for $g \equiv g(\xi)$:

$$g = 1 + \frac{k}{2h^2}\left(e^{-i\xi h} - 2 + e^{i\xi h}\right)(1+g)$$

and hence

$$g = \frac{1 + \frac{1}{2}z}{1 - \frac{1}{2}z} \tag{12.24}$$

where

$$\begin{aligned} z &= \frac{k}{h^2}(e^{-i\xi h} - 2 + e^{i\xi h}) \\ &= \frac{2k}{h^2}(\cos(\xi h) - 1). \end{aligned} \tag{12.25}$$

Since $z \le 0$ for all ξ, we see that $|g| \le 1$ and the method is stable for any choice of k and h.

Note that (12.24) agrees with the root ζ_1 found for the Trapezoidal method in Example 8.6, while the z determined in (12.25), for certain values of ξ, is simply k times an eigenvalue λ_p from (12.13), the eigenvalues of the Method of Lines matrix. So there is a close connection between the von Neumann approach and the MOL reduction to a system of ODE's.

12.7 Multi-dimensional problems

In two space dimensions the heat equation takes the form

$$u_t = u_{xx} + u_{yy} \tag{12.26}$$

with initial conditions $u(x,y,0) = \eta(x,y)$ and boundary conditions all along the boundary of our spatial domain Ω. We can discretize in space using a discrete Laplacian of the form considered in Chapter 3, say the five-point Laplacian from Section 3.2:

$$\nabla_h^2 U_{ij} = \frac{1}{h^2}(U_{i-1,j} + U_{i+1,j} + U_{i,j-1} + U_{i,j+1} - 4U_{ij}). \tag{12.27}$$

If we then discretize in time using the trapezoidal method, we will obtain the two-dimensional version of the Crank-Nicolson method,

$$U_{ij}^{n+1} = U_{ij}^n + \frac{k}{2}[\nabla_h^2 U_{ij}^n + \nabla_h^2 U_{ij}^{n+1}]. \tag{12.28}$$

Since this method is implicit, we must solve a system of equations for all the U_{ij} where the matrix has the same nonzero structure as for the elliptic systems considered in Chapters 3 and 5. This matrix is large and sparse, and we generally do not want to solve the system by a direct method such as Gaussian Elimination. In fact this is even more true for the systems we are now considering than for the elliptic equation, because of the slightly different nature of this system, which makes other approaches even more efficient relative to direct methods. It is also extremely important now that we use the most efficient method possible, because we must now solve a linear system of this form *in every time step*, and we may need to take thousands of time steps to solve the time-dependent problem.

We can rewrite the equations (12.28) as

$$\left(I - \frac{k}{2}\nabla_h^2\right) U_{ij}^{n+1} = \left(I + \frac{k}{2}\nabla_h^2\right) U_{ij}^n. \tag{12.29}$$

The matrix for this linear system has the same pattern of nonzeros as the matrix for ∇_h^2 (see Chapter 3), but the values are scaled by $k/2$ and then subtracted from the identity matrix, so that the diagonal elements are fundamentally different. If we call this matrix A,

$$A = I - \frac{k}{2}\nabla_h^2,$$

then we find that the eigenvalues of A are

$$\lambda_{p,q} = 1 - \frac{k}{h^2}\left[(\cos(p\pi h) - 1) + (\cos(q\pi h) - 1)\right]$$

for $p,\ q = 1,\ 2,\ \ldots,\ m$, where we have used the expression for the eigenvalues of ∇_h^2 from Section 3.3. Now the largest eigenvalue of the matrix A thus has magnitude $O(k/h^2)$ while the ones closest to the origin are at $1 + O(k)$. As a result the condition number of A is $O(k/h^2)$. By contrast, the discrete Laplacian ∇_h^2 alone has condition number $O(1/h^2)$ as we found in Section 3.3. The smaller condition number in the present case can be expected to lead to faster convergence of iterative methods.

Moreover, we have an excellent starting guess for the solution U^{n+1} to (12.28), a fact that we can use to good advantage with iterative methods but not with direct methods. Since $U_{ij}^{n+1} = U_{ij}^n + O(k)$, we can use U_{ij}^n, the values from the previous time step, as initial values $U_{ij}^{[0]}$ for an iterative method. We might do even better by extrapolating forward in time, using say $U_{ij}^{[0]} = 2U_{ij}^n - U_{ij}^{n-1}$, or by using an explicit method, say

$$U_{ij}^{[0]} = (I + k\nabla_h^2)U_{ij}^n.$$

This explicit method (forward Euler) would probably be unstable as a time-marching procedure if we used only this with the value of k we have in mind, but it can be used successfully as a way to generate initial data for an iterative procedure.

Because of the combination of a reasonably well-conditioned system and very good initial guess, we can often get away with taking only one or two iterations in each time step, and still get global second order accuracy.

12.8 The LOD method

Rather than solving the coupled sparse matrix equation for all the unknowns on the grid simultaneously as in (12.29), an alternative approach is to replace this fully-coupled single time step by a sequence of steps, each of which is coupled in only one space direction, resulting in a set of tridiagonal systems which can be solved much more easily. One example is the *Locally One-Dimensional (LOD)* method:

$$U_{ij}^* = U_{ij}^n + \frac{k}{2}(D_x^2 U_{ij}^n + D_x^2 U_{ij}^*) \tag{12.30}$$

$$U_{ij}^{n+1} = U_{ij}^* + \frac{k}{2}(D_y^2 U_{ij}^* + D_y^2 U_{ij}^{n+1}). \tag{12.31}$$

or, in matrix form,

$$\left(I - \frac{k}{2}D_x^2\right)U^* = \left(I + \frac{k}{2}D_x^2\right)U^n \tag{12.32}$$

$$\left(I - \frac{k}{2}D_y^2\right)U^{n+1} = \left(I + \frac{k}{2}D_y^2\right)U^*. \tag{12.33}$$

In (12.30) we apply Crank-Nicolson in the x-direction only, solving $u_t = u_{xx}$ alone over time k, and we call the result U^*. Then in (12.31) we take this result and apply Crank-Nicolson in the y-direction to it, solving $u_t = u_{yy}$ alone, again over time k. Physically this corresponds to modeling diffusion in the x- and y-directions over time k as a decoupled process in which we first allow u to diffuse only in the x-direction and then only in the y-direction. If the time steps are very short then this might be expected to give similar physical behavior and hence convergence to the correct behavior as $k \to 0$. In fact, for the constant coefficient diffusion problem, it can even be shown that (in the absence of boundaries at least) this alternating diffusion approach gives *exactly* the same behavior as the original two-dimensional diffusion. Diffusing first in x alone over time k and then in y alone over time k gives the same result as if the diffusion occurs simultaneously in both directions.

Numerically there is a great advantage in using (12.32) and (12.33) rather than the fully coupled (12.29). In (12.32) the unknowns U^*_{ij} are coupled together only across each row of the grid. For any fixed value of j we have a *tridiagonal system* of equations to solve for $U^*_{ij}(i = 1, 2, \ldots, m)$. The system obtained for each value of j is completely decoupled from the system obtained for other values of j. Hence we have a set of $m+2$ tridiagonal systems to solve (for $j = 0, 1, \ldots, m+1$), each of dimension m, rather than a single coupled system with m^2 unknowns as in (12.29). Since each of these systems is tridiagonal, it is easily solved in $O(m)$ operations by Gaussian elimination and there is no need for iterative methods. (In the next section we will see why we need to solve these for $j = 0$ and $j = m+1$ as well as at the interior grid points.)

Similarly, (12.31) decouples into a set of m tridiagonal systems in the y-direction for $i = 1, 2, \ldots, m$. Hence taking a single time step requires solving $2m+2$ tridiagonal systems of size m, and thus $O(m^2)$ work. Since there are m^2 grid points, this is the optimal order and no worse than an explicit method, except for a constant factor.

12.8.1 Boundary conditions

In solving the second set of systems (12.31), we need boundary values U^*_{i0} and U^{n+1}_{i0} along the bottom boundary and $U^*_{i,m+1}$ and $U^{n+1}_{i,m+1}$ along the top boundary, for terms that go on the right-hand side of each tridiagonal system. The values at level $n+1$ are available from the given boundary data for the heat equation, by evaluating the boundary conditions at time t_{n+1} (assuming Dirichlet boundary conditions are given). To obtain the values U^*_{i0} we solve equation (12.30) for $j = 0$ and $j = m+1$ (along the boundaries) in addition to the systems along each row interior to the grid.

In order to solve the first set of systems (12.30), we need boundary values U^n_{0j} and U^*_{0j} along the left boundary and values $U^n_{m+1,j}$ and $U^*_{m+1,j}$ along the right boundary. The values at level n come from the given boundary conditions, but we must determine the intermediate boundary conditions at level $*$ along these boundaries. It is not immediately clear what values should be used. One might be tempted to think of level $*$ as being half way between t_n and t_{n+1}, since U^* is generated in the middle of the two-step procedure used to obtain U^{n+1} from U^n. If this were valid, then evaluating the given boundary data at time $t_{n+1/2} = t_n + k/2$ might provide values for U^* on the boundary. This is not a good idea, however, and would lead to a degredation of accuracy. The problem is that in the first step, equation (12.30) does not model the full heat equation over time $k/2$, but rather models part of the equation (diffusion in x alone) over the full time step k. The values along the boundary will in general evolve quite differently in the two different cases.

To determine proper values for U^*_{0j} and $U^*_{m+1,j}$, we can use the equations (12.31) along the left and right boundaries. At $i = 0$, for example, this equation gives a system of equations along the left boundary that can be viewed as a tridiagonal linear system or the unknowns U^*_{0j} in terms of the values

U_{0j}^{n+1}, which are already known from the boundary conditions at time t_{n+1}. Note that we are solving this equation backwards from the way it will be used in the second step of the LOD process on the interior of the grid, and this works only because we already know U_{0j}^{n+1} from boundary data.

Since we are solving this equation backwards, we can view this as solving the diffusion equation $u_t = u_{yy}$ over a time step of length $-k$, backwards in time. This makes sense physically — the intermediate solution U^* represents what is obtained from U^n by doing diffusion in x alone, with no diffusion yet in y. There are in principle two ways to get this, either by starting with U^n and diffusing in x, or by starting with U^{n+1} and "undiffusing" in y. We are using the latter approach along the boundaries to generate data for U^*.

Equivalently we can view this as solving the *backward heat equation* $u_t = -u_{yy}$ over time k. This may be cause for concern, since the backward heat equation is ill-posed. However, since we are only doing this over one time step starting with given values U_{0j}^{n+1} in each time step, this turns out to be a stable procedure.

There is still a difficulty at the corners. In order to solve (12.31) for U_{0j}^*, $j = 1, 2, \ldots, m$, we need to know the values of U_{00}^* and $U_{0,m+1}^*$ that are the boundary values for this system. These can be approximated using some sort of explicit and uncentered approximation to either $u_t = u_{xx}$ starting with U^n, or to $u_t = -u_{yy}$ starting with U^{n+1}. For example we might use

$$U_{00}^* = U_{00}^{n+1} - \frac{k}{h^2}(U_{00}^{n+1} - 2U_{01}^{n+1} + U_{02}^{n+1}),$$

which uses the approximation to u_{yy} centered at (x_0, y_1).

Alternatively, rather than solving the tridiagonal systems obtained from (12.31) for U_{0j}^*, we could simply use an explicit approximation to the backwards heat equation along this boundary,

$$U_{0j}^* = U_{0j}^{n+1} - \frac{k}{h^2}(U_{0,j-1}^{n+1} - 2U_{0j}^{n+1} + U_{0,j+1}^{n+1}), \tag{12.34}$$

for $j = 1, 2, \ldots, m$. This eliminates the need for values of U^* in the corners. Again, since this is not iterated but only done starting with given (and presumably smooth) boundary data U^{n+1} in each time step, this yields a stable procedure.

12.8.2 Accuracy and stability

With proper treatment of the boundary conditions, it can be shown that the LOD method is second order accurate. It can also be shown that this method, like full Crank-Nicolson, is unconditionally stable for any time step.

12.8.3 The ADI method

A modification of the LOD method is also often used, in which the two steps each involve discretization in only one spatial direction at the advanced time level (giving decoupled tridiagonal systems again), but coupled with discretization in the *opposite* direction at the old time level. The classical method of this form is:

$$\begin{aligned} U_{ij}^* &= U_{ij}^n + \frac{k}{2}(D_y^2 U_{ij}^n + D_x^2 U_{ij}^*) && (12.35)\\ U_{ij}^{n+1} &= U_{ij}^* + \frac{k}{2}(D_x^2 U_{ij}^* + D_y^2 U_{ij}^{n+1}). && (12.36)\end{aligned}$$

This is called the *Alternating Direction Implicit (ADI)* method and was first introduced by Douglas and Rachford [DR56]. This again gives decoupled tridiagonal systems to solve in each step:

$$\begin{aligned} \left(I - \frac{k}{2}D_x^2\right)U^* &= \left(I + \frac{k}{2}D_y^2\right)U^n && (12.37)\\ \left(I - \frac{k}{2}D_y^2\right)U^{n+1} &= \left(I + \frac{k}{2}D_x^2\right)U^*. && (12.38)\end{aligned}$$

With this method, each of the two steps involves diffusion in both the x- and y-directions. In the first step the diffusion in x is modelled implicitly while diffusion in y is modelled explicitly, with the roles reversed in the second step. In this case each of the two steps can be shown to give a *first-order accurate* approximation to the full heat equation over time $k/2$, so that U^* represents a first-order accurate approximation to the solution at time $t_{n+1/2}$. Because of the symmetry of the two steps, however, the local error introduced in the second step almost exactly cancels the local error introduced in the first step, so that the combined method is in fact *second-order accurate* over the full time step.

Because U^* does approximate the solution at time $t_{n+1/2}$ in this case, it is possible to simply evaluate the given boundary conditions at time $t_{n+1/2}$ to generate the necessary boundary values for U^*. This will maintain second-order accuracy. A better error constant can be achieved by using slightly modified boundary data which introduces the expected error in U^* into the boundary data that should be cancelled out by the second step.

12.9 Exercises

Exercise 12.1 *Compute the dominant term in the truncation error of Crank-Nicolson.*

part

$\boxed{4B}$

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.

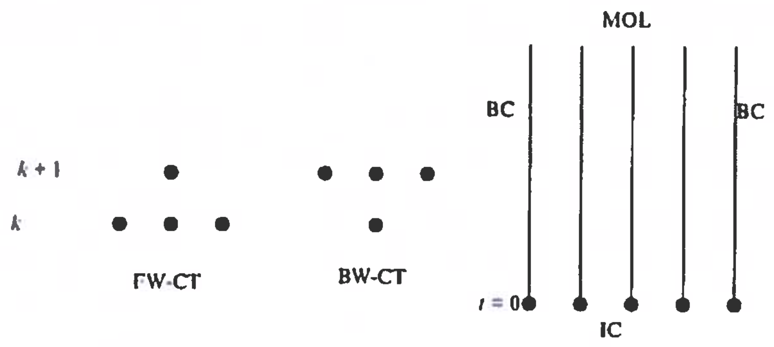

Fig. 4.1. Diagram of the finite difference stencil for the forward and backward Euler methods, and the MOL.

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,

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

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

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

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

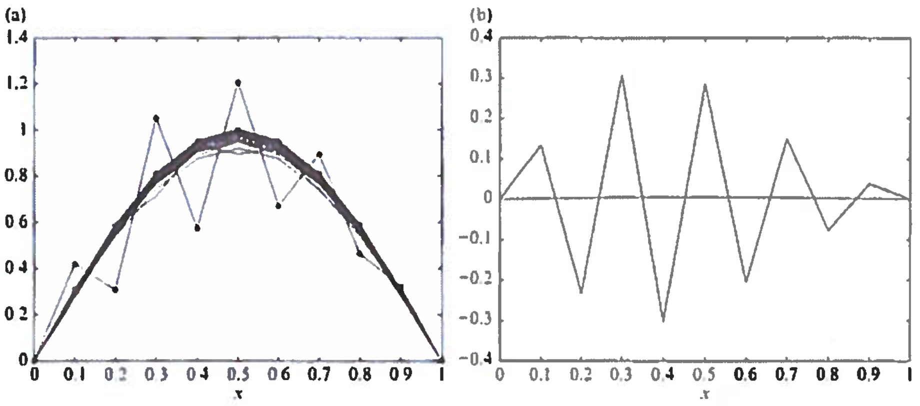

Fig. 4.2. (a) Plot of the computed solutions using two different time step sizes and the exact solution at some time for the test problem. (b) Error plots of the computed solution using the two different step sizes: one is stable and the error is small; the other one is unstable and the error grows rapidly.

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

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

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 $k + 1$:

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

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

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 **c** is a vector. Both A and **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 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

$$\frac{U_i^{k+1} - U_i^k}{\Delta t} = \frac{\beta_{i-\frac{1}{2}}^k U_{i-1}^k - (\beta_{i-\frac{1}{2}}^k + \beta_{i+\frac{1}{2}}^k)U_i^k + \beta_{i+\frac{1}{2}}^k U_{i+1}^k}{2h^2}$$

$$+ \frac{\beta_{i-\frac{1}{2}}^{k+1} U_{i-1}^{k+1} - (\beta_{i-\frac{1}{2}}^{k+1} + \beta_{i+\frac{1}{2}}^{k+1})U_i^{k+1} + \beta_{i+\frac{1}{2}}^{k+1} U_{i+1}^{k+1}}{2h^2} + \frac{1}{2}\left(f_i^k + f_i^{k+1}\right). \quad (4.13)$$

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 $0 < \theta \leq 1/2$, 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

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

Then as the Fourier transform and its inverse are unique, $\widehat{\partial u/\partial x} = i\omega\hat{u}$. 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

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

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,

$$\widehat{u}_t + \widehat{au_x} = 0, \quad \text{or} \quad \widehat{u}_t + ai\omega\hat{u} = 0, \qquad \hat{u}(\omega,0) = \hat{u}_0(\omega)$$

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

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

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

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

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 $x_i = ih$, 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

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

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:

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

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 $O\big((\Delta t)^2+h^2\big)$. 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

$$\frac{U_{ij}^{k+\frac12}-U_{ij}^{k}}{(\Delta t)/2}=\frac{U_{i-1,j}^{k+\frac12}-2U_{ij}^{k+\frac12}+U_{i+1,j}^{k+\frac12}}{h_x^2}+\frac{U_{i,j-1}^{k}-2U_{ij}^{k}+U_{i,j+1}^{k}}{h_y^2}+f_{ij}^{k+\frac12},$$
$$\frac{U_{ij}^{k+1}-U_{ij}^{k+\frac12}}{(\Delta t)/2}=\frac{U_{i-1,j}^{k+\frac12}-2U_{ij}^{k+\frac12}+U_{i+1,j}^{k+\frac12}}{h_x^2}+\frac{U_{i,j-1}^{k+1}-2U_{ij}^{k+1}+U_{i,j+1}^{k+1}}{h_y^2}+f_{ij}^{k+\frac12},$$
$$(4.50)$$

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

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

We can go further to get

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

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

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

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,

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

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 $|g(\xi_1, \xi_2)| \le 1$, 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 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}:

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

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

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

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

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