Chapter 3

The Diffusion Equation

3.1 Forward Time Centred Space (FTCS) Scheme

In this chapter we shall focus on methods of solving the diffusion equation with source term:

$$\frac{\partial v}{\partial t} = D\frac{\partial^2 v}{\partial x^2} + f(x,t), \tag{3.1}$$

in the domain $(x,t) \in [x_L, x_R] \times [0,T)$. This is to be solved with initial data $v(x,0) = V(x)$ and boundary conditions specified at $x = x_L$ and $x = x_R$. We shall denote the size of the spatial domain by $L = v_R - v_L$.

The similarity solutions which we constructed in Chap. 2 do not generalise to cases with a finite domain, general initial condition or source term (even though they may exhibit behaviour which is typical in an asymptotic sense). We need a more robust method. This leads us to the topic of numerical PDE's proper. We used finite difference approximations of derivatives to build numerical algorithms capable of solving ODEs. We shall do the same here although life is complicated a bit by the fact that we now have derivatives with respect to space and time.

Let us discretise space first, ignoring the source term in Eq. (3.1) for the time being. We divide the spatial domain into $N-1$ intervals of length $\Delta x = \frac{L}{N-1}$ using N equally spaced points, $x_i = x_L + i\Delta x$. With this definition, $x_0 = x_L$ and $x_{N-1} = x_R$. We create a vector, $\mathbf{v}(t) \in \mathbb{R}^N$ from the values, $v_i(t) = v(x_i,t)$, of $v(x,t)$ at the N grid points, x_i:

$$\mathbf{v}(t) = (v_0(t), v_1(t), \ldots, v_{N-1}(t)).$$

Eq. (3.1) tells us how each component of $\mathbf{v}$ evolves in time:

$$\frac{\partial v_i}{\partial t}(t) = D\frac{\partial^2 v}{\partial x^2}(x_i,t).$$

We can approximate the second derivative on the RHS at a given time with a finite difference:

$$\frac{\partial v_i}{\partial t}(t) = \frac{D}{(\Delta x)^2}\left[v_{i+1}(t) - 2v_i(t) + v_{i-1}(t)\right] + O(\Delta x).$$

We are immediately confronted with the question of what to do at the boundaries. For now lets assume periodic boundary conditions (the solution wraps around at the end of the spatial domain):

$$\begin{aligned} v_N(t) &= v_0(t) \\ v_{-1}(t) &= v_{N-1}(t). \end{aligned}$$

We will return to the question of how to handle other boundary conditions in Sec. 3.2. Note that we now have an approximation of the form

$$\frac{d\mathbf{v}}{dt} = \mathbf{G}(\mathbf{v}). \tag{3.2}$$

Although the dimension, N, of this first order system is considerably larger in practice than those which we solved in Chap. 1, in principle everything we learned there is applicable now. It is clear that $\mathbf{G}(\mathbf{v})$ is rather simple here:

$$\mathbf{G}(\mathbf{v}) = A\mathbf{v} \tag{3.3}$$

where is an $N \times N$ matrix of the form (for $N = 5$):

$$A = \frac{D}{(\Delta x)^2} \begin{pmatrix} -2 & 1 & 0 & 0 & 1 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 1 & -2 & 1 & 0 \\ 0 & 0 & 1 & -2 & 1 \\ 1 & 0 & 0 & 1 & -2 \end{pmatrix} \tag{3.4}$$

With the exception of the effects of the periodic boundary conditions, A, is a tridiagonal matrix. Tridiagonal matrices and banded matrices in general are attractive in designing numerical algorithms since there exist very efficient algorithms for performing matrix operations with such matrices.

To solve our original problem we must now select a timestepping algorithm to advance Eq. (3.2). The simplest choice would be to use the Forward Euler Method. We choose a time increment, h, and create temporal gridpoints, $t_j = j\,h$. The Forward Euler Scheme is

$$\mathbf{v}(t_{j+1}) = \mathbf{v}(t_j) + hA\mathbf{v}(t_j) \tag{3.5}$$

If we now adopt the notation

$$v_{i,j} = v(x_i, t_j).$$

we can see that the Forward Euler Scheme gives us an explicit timestepping algorithm for Eq. (3.1) without the source term:

$$v_{i,j+1} = v_{i,j} + \delta\left[v_{i+1,j} - 2v_{i,j} + v_{i-1,j}\right] \tag{3.6}$$

where

$$\delta = \frac{Dh}{(\Delta x)^2}. \tag{3.7}$$

Eq. (3.6) is called the Forward Time Centred Space (FTCS) algorithm. It can be written as a simple matrix multiplication:

$$\mathbf{v}_{j+1} = B\mathbf{v}_j \tag{3.8}$$

where

$$B = \begin{pmatrix} 1-2\delta & \delta & 0 & 0 & \delta \\ \delta & 1-2\delta & \delta & 0 & 0 \\ 0 & \delta & 1-2\delta & \delta & 0 \\ 0 & 0 & \delta & 1-2\delta & \delta \\ \delta & 0 & 0 & \delta & 1-2\delta \end{pmatrix}. \tag{3.9}$$

For this reason, there are strong links between linear algebra and numerical analysis of PDEs. One of the main differences between the algorithms which we studied in Chap 1 and Eq. (3.6) is that there are now two sources of approximation - a spatial and temporal discretisation. These can interact to make questions of stability and convergence more delicate.

3.2 Source Terms and Boundary Conditions in the FTCS Scheme

3.2.1 Including a source term in the FTCS scheme

It is not hard to incorporate the source term in Eq. (3.1) into the FTCS scheme. We create vector, $\mathbf{f}(t) \in \mathbb{R}^N$, by evaluating the source term at each spatial gridpoint:

$$\mathbf{f}(t) = (f_0(t), f_1(t), \ldots, f_{N-1}(t)),$$

where $f_i(t) = f(x_i, t)$. The analogue of Eq. (3.2) is then

$$\frac{d\mathbf{v}}{dt} = A(\mathbf{v}) + \mathbf{f}(t). \tag{3.10}$$

with the matrix A still given by Eq. (3.4). Note that the problem is now potentially non-autonomous. We could increase the dimension by one to get an autonomous system as we learned to do in Chap. 1. If we did that, however, the **G** operator for the augmented system would no longer be banded or (unless the source was very special) linear. This would be bad news for large N since we would no longer be able to use all those fast algorithms for performing linear operations on banded matrices. Instead, for PDE applications, we generally prefer to work directly with the non-autonomous system and write our Forward Euler Scheme with the source included explicitly:

$$\mathbf{v}(t_{j+1}) = \mathbf{v}(t_j) + hA\mathbf{v}(t_j) + h\mathbf{f}(t_j). \tag{3.11}$$

A single timestep of the FTCS algorithm, Eq. (3.8), now requires a vector addition in addition to a matrix multiplication:

$$\mathbf{v}_{j+1} = B\mathbf{v}_j + \mathbf{b}_j \tag{3.12}$$

where $\mathbf{b}_j = h\mathbf{f}_j$. One thing to watch out for here is not to use more sophisticated integrators which have been designed for non-autonomous systems.

3.2.2 Imposing Dirichlet Boundary Conditions in the FTCS scheme

Dirichlet conditions:

$$\begin{aligned} v(x_L, t) &= D_L(t) \\ v(x_R, t) &= D_R(t). \end{aligned} \tag{3.13}$$

(D_L and D_R could be constant or zero!) With these boundary conditions we know the time evolution of v_0 and v_{N-1}. Thus we only need to compute the solution for the points on the interior of the domain, $[x_L + \Delta x, x_R - \Delta x]$. From a conceptual point of view, it is easier to think of $[x_L + \Delta x, x_R - \Delta x] = [\widetilde{x}_L.\widetilde{x}_R]$ as a new domain to be discretised. The boundary points are considered to be external to the domain (sometimes called "false points"). Thus we divide the new domain $[\widetilde{x}_L.\widetilde{x}_R]$ into $\widetilde{N}-1 = N-3$ intervals of length Δx using $\widetilde{N} = N-2$ equally spaced points $x_i = \widetilde{x}_L + i\Delta x$. In the new domain, the boundary conditions are applied at nodes x_{-1} and $x_{\widetilde{N}}$.

The FTCS scheme is the same as before for the interior points $i = 1, ldots\widetilde{N} - 2$:

$$v_{i,j+1} = v_{i,j} + \delta\left[v_{i+1,j} - 2v_{i,j} + v_{i-1,j}\right] \tag{3.14}$$

but at the boundary points, $i = 0$ and $i = \widetilde{N} - 1$ we have

$$\begin{aligned} v_{0,j+1} &= v_{0,j} + \delta\left[v_{1,j} - 2v_{0,j} + D_L(t_j)\right] \\ v_{\widetilde{N}-1,j+1} &= v_{\widetilde{N}-1,j} + \delta\left[D_R(t_j) - 2v_{\widetilde{N}-1,j} + v_{\widetilde{N}-2,j}\right]. \end{aligned} \tag{3.15}$$

Eqs. (3.14) and (3.15) are equivalent to the $\widetilde{N} \times \widetilde{N}$ linear system

$$\mathbf{v}_{j+1} = B\mathbf{v}_j + \mathbf{b}_j \tag{3.16}$$

where (for $\widetilde{N} = 5$)

$$B = \begin{pmatrix} 1-2\delta & \delta & 0 & 0 & 0 \\ \delta & 1-2\delta & \delta & 0 & 0 \\ 0 & \delta & 1-2\delta & \delta & 0 \\ 0 & 0 & \delta & 1-2\delta & \delta \\ 0 & 0 & 0 & \delta & 1-2\delta \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} \delta D_L(t_j) \\ 0 \\ 0 \\ 0 \\ \delta D_R(t_j) \end{pmatrix}. \tag{3.17}$$

3.2.3 Imposing Neumann Boundary Conditions in the FTCS scheme

Neumann conditions:

$$\begin{aligned} \frac{\partial v}{\partial x}(x_L,t) &= N_L(t) \\ \frac{\partial v}{\partial x}(x_R,t) &= N_R(t). \end{aligned} \tag{3.18}$$

To implement these boundary conditions, we again use "false points", x_{-1} and x_N which are external points. We use a centred difference to approximate $\frac{\partial v}{\partial x}(x_L,t)$ and set it equal to the desired boundary condition:

$$\frac{\partial v}{\partial x}(x_L,t) = \frac{v_1 - v_{-1}}{2\Delta x} + O(\Delta x^2) = N_L(t).$$

From this we can determine v_{-1}:

$$v_{-1}(t) = v_1(t) - 2\,\Delta x\, N_L(t). \tag{3.19}$$

Similarly at the right boundary we determine v_N:

$$v_N(t) = v_{N-1}(t) + 2\,\Delta x\, N_R(t). \tag{3.20}$$

The FTCS scheme is the same as before for the interior points $i = 1, ldots\widetilde{N} - 2$:

$$v_{i,j+1} = v_{i,j} + \delta\,[v_{i+1,j} - 2v_{i,j} + v_{i-1,j}] \tag{3.21}$$

but at the boundary points, $i = 0$ and $i = N-1$ we have

$$\begin{aligned} v_{0,j+1} &= v_{0,j} + \delta\,[2v_{1,j} - 2v_{0,j}] - \frac{2Dh}{\Delta x} N_L(t_j) \\ v_{N-1,j+1} &= v_{N-1,j} + \delta\,[-2v_{N-1,j} + 2v_{N-2,j}] + \frac{2Dh}{\Delta x} N_R(t_j). \end{aligned} \tag{3.22}$$

Eqs. (3.21) and (3.22) are equivalent to the $N \times N$ linear system

$$\mathbf{v}_{j+1} = B\mathbf{v}_j + \mathbf{b}_j \tag{3.23}$$

where (for $N = 5$)

$$B = \begin{pmatrix} 1-2\delta & 2\delta & 0 & 0 & 0 \\ \delta & 1-2\delta & \delta & 0 & 0 \\ 0 & \delta & 1-2\delta & \delta & 0 \\ 0 & 0 & \delta & 1-2\delta & \delta \\ 0 & 0 & 0 & 2\delta & 1-2\delta \end{pmatrix} \qquad \mathbf{b} = \begin{pmatrix} -\frac{2Dh}{\Delta x} N_L(t_j) \\ 0 \\ 0 \\ 0 \\ \frac{2Dh}{\Delta x} N_R(t_j) \end{pmatrix}. \tag{3.24}$$

Fig. 3.1 shows some numerical solutions to the diffusion equation with gaussian initial conditions obtained using the FTCS scheme. Although Dirichlet boundary conditions have been imposed, Fig. 3.1 shows the evolution at early times before the solution starts to feel the boundaries. The solution is therefore very well approximated by the self similar solution, Eq. (2.48), obtained in Chap. 2. The solution on the left has a timestep of $h = 2.00 \times 10^{-2}$. The solution on the right has a timestep of $h = 2.66 \times 10^{-2}$. It is clear that something goes catastrophically wrong with the FTCS scheme under certain circumstances. The oscillatory behaviour captured in the right hand frame of Fig. 3.1 is the leading edge of an exponentially growing instability which, within a few more timesteps completely engulfs the entire solution rendering the numerical solution useless.

Is this instability absent in the left hand frame or is would the left hand computation fall victim to the same instability if we waited for slightly longer time? The question of when the numerical solution obtained from a given numerical algorithm converges to the exact solution is one of the central questions of numerical analysis. We now turn to this issue.

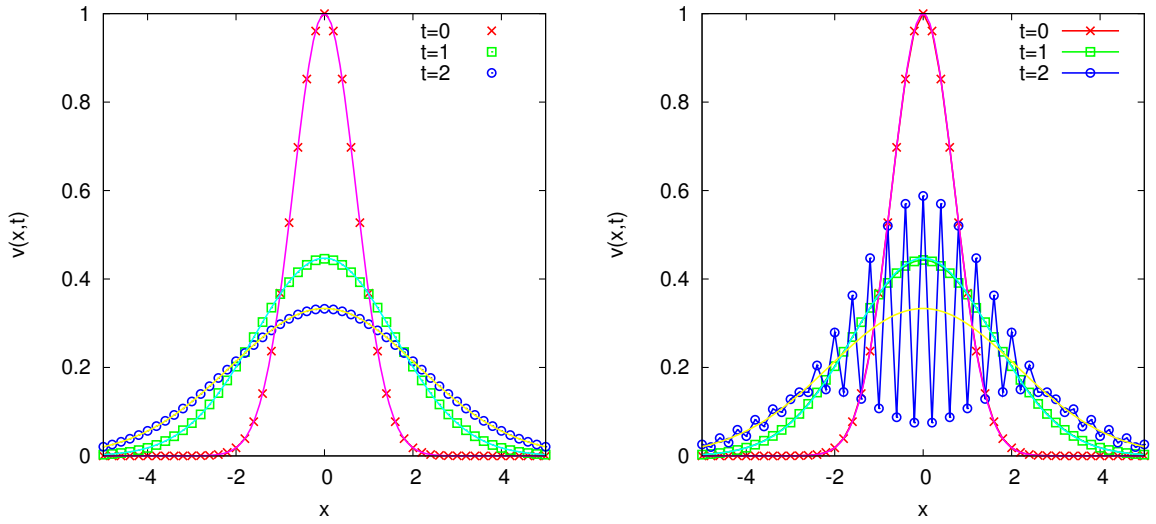

Figure 3.1: Snapshots of the numerical solution of Eq. (3.1) with $f = 0$ and $D = 1$ computed using the FTCS scheme, Eq. (3.6) . The spatial interval was $[-20 : 20]$ (only the range $[-5 : 5]$ is shown) with Dirichlet boundary conditions $v(-20, t) = v(20, t) = 0$ imposed at the edges. The initial condition was $v(x, 0) = \mathrm{e}^{-x^2}$. For the computation on the left, $h = 0.02$. For the computation on the right, $h = 0.0266$. Solid lines show the approximate analytic solution.

3.3 Consistency and Stability of the FTCS Scheme

We now look at the concepts of consistency and stability which allow us to understand when a numerical solution to a PDE converges to the exact solution. We shall then apply them to the FTCS scheme.

Consistency:

a finite difference scheme is *consistent* if the numerical solution computed after a fixed number of steps converges to the exact solution as h and Δx tend to zero.

Consistency ensures that the finite difference equation converges to the original PDE.

Stability:

a finite difference scheme is *stable* if the numerical solution computed after a fixed time remains bounded as $h \to 0$.

Stability ensures that the numerical solution at a finite time does not blow up as the timestep is reduced to zero. Consistency and stability together ensure convergence of the numerical solution according to the following theorem:

Lax–Richtmeyer Theorem:

a finite difference approximation to a well posed linear IVP converges to the exact solution as h and Δx tend to zero if and only if it is consistent and stable.

3.3.1 Consistency of the FTCS Scheme

We want to compare the numerical and exact solution after a fixed number of timesteps, say $j+1$. Let us denote the numerically computed solution at space-time point x_i, t_j by $\widetilde{v}_{i,j}$ to distinguish it from corresponding value of the exact solution of Eq. (3.1), which we denote by $v_{i,j}$. For simplicity, we shall assume that $f = 0$. Recall how $\widetilde{v}_{i,j+1}$ is computed using the FTCS scheme:

$$\widetilde{v}_{i,j+1} = \delta\widetilde{v}_{i-1,j} + (1-2\delta)\widetilde{v}_{i,j} + \delta\widetilde{v}_{i+1,j}, \tag{3.25}$$

where $\delta = \frac{Dh}{(\Delta x)^2}$. The error at each point is

$$\varepsilon_{i,j} = v_{i,j} - \widetilde{v}_{i,j}.$$

From Eq. (3.25), we easily obtain

$$\varepsilon_{i,j+1} = \delta(\varepsilon_{i-1,j} + \varepsilon_{i+1,j}) + (1-2\delta)\varepsilon_{i,j} + \delta(v_{i-1,j} + v_{i+1,j}) + (1-2\delta)v_{i,j} - v_{i,j+1}. \tag{3.26}$$

By Taylor's Theorem

$$\begin{aligned} v_{i+1,j} &= v_{i,j} + (\Delta x)\frac{\partial v}{\partial x}(x_i,t_j) + \frac{1}{2}(\Delta x)^2\frac{\partial^2 v}{\partial x^2}(\eta_i,t_j) \\ v_{i-1,j} &= v_{i,j} - (\Delta x)\frac{\partial v}{\partial x}(x_i,t_j) + \frac{1}{2}(\Delta x)^2\frac{\partial^2 v}{\partial x^2}(\mu_i,t_j) \\ v_{i,j+1} &= v_{i,j} + h\frac{\partial v}{\partial t}(x_i,\tau_j), \end{aligned}$$

for some $\eta_i \in [x_i, x_{i+1}]$, $\mu_i \in [x_{i-1}, x_i]$ and $\tau_j \in [t_j, t_{j+1}]$. Putting these into Eq. (3.26) and doing some algebra we obtain

$$\varepsilon_{i,j+1} = \delta(\varepsilon_{i-1,j} + \varepsilon_{i+1,j}) + (1-2\delta)\varepsilon_{i,j} + h\left[\frac{D}{2}\left(\frac{\partial^2 v}{\partial x^2}(\eta_i,t_j) + \frac{\partial^2 v}{\partial x^2}(\mu_i,t_j)\right) - \frac{\partial v}{\partial t}(x_i,\tau_j)\right]. \tag{3.27}$$

Now, the solution of the PDE exists so the absolute value of term in the square brackets has a maximum value over all the (i,j) in the computational grid which we denote by $\mathcal{M}$:

$$\mathcal{M} = \max_{i,j}\left|\frac{D}{2}\left(\frac{\partial^2 v}{\partial x^2}(\eta_i,t_j) + \frac{\partial^2 v}{\partial x^2}(\mu_i,t_j)\right) - \frac{\partial v}{\partial t}(x_i,\tau_j)\right|. \tag{3.28}$$

Let us denote by E_j the maximum over the spatial points of absolute value of the error at a fixed time-slice, j:

$$E_j = \max_i |\varepsilon_{i,j}|. \tag{3.29}$$

From Eq. (3.27), provided that $1-2\delta > 0$,

$$\begin{aligned} |\varepsilon_{i,j+1}| &\leq \delta\left[E_j + E_j\right] + (1-2\delta)E_j + \mathcal{M}h \\ &= E_j + h\mathcal{M}. \end{aligned}$$

Let us now assume that $1-2\delta > 0$ and take the maximum over i:

$$E_{j+1} \leq E_j + h\mathcal{M}. \tag{3.30}$$

We can iterate this argument:

$$E_{j+1} \leq E_j + h\mathcal{M} \leq E_{j-1} + 2h\mathcal{M} \ldots \leq E0 + jh\mathcal{M} = jh\mathcal{M}, \tag{3.31}$$

since $E_0 = 0$. Now let $h \to 0$ and $\Delta x \to 0$. E_{j+1} will tend to zero provided that $\mathcal{M}$ remains finite. Clearly as $h \to 0$ and $\Delta x \to 0$, $\eta_i \to x_i$, $\mu_i \to x_i$ and $\tau_j \to t_j$. Then from Eq. (3.28) and Eq. (3.1) (recall we are taking $f = 0$) we see that $\mathcal{M} \to 0$. Hence the error after $j + 1$ steps, E_{j+1} tends to zero. We conclude that the FTCS scheme, Eq. (3.25), is consistent if

$$\delta = \frac{Dh}{(\Delta x)^2} < \frac{1}{2}. \tag{3.32}$$

Notice that in this argument, we cannot take h and Δx to zero independently.

3.3.2 Stability of the FTCS Scheme

Reminder: Finding the largest eigenvalue of a matrix:
If A is an $N \times N$ diagonalisable matrix with eiganvalues $\lambda_1 > \lambda_2 \geq \lambda_3 \geq \ldots \geq \lambda_N$. The for any vector $\mathbf{b}_0$ having nonzero component in the direction of the eigenvector associated with λ_1, (iteratively) define

$$\mathbf{b}_{k+1} = \frac{A\,\mathbf{b}_k}{|A\,\mathbf{b}_k|}.$$

Then $\mathbf{b}_{k+1} \to \lambda_1 \mathbf{b}_k$ as $k \to \infty$. This is called the Power Method.

Let us consider the case of Dirichlet conditions for concreteness. Eq. (3.16) shows that the FTCS scheme, Eq. (3.25), is naturally expressed as a linear system,

$$\widetilde{\mathbf{v}}_{j+1} = B\widetilde{\mathbf{v}}_j + \mathbf{b}_j.$$

In reality, we have some error relative to the exact solution which means

$$\begin{aligned} & \mathbf{v}_{j+1} + \varepsilon_{j+1} = B(\mathbf{v}_j + \varepsilon_j) + \mathbf{b}_j \\ \Rightarrow \quad & \varepsilon_{j+1} = B\varepsilon_j \\ \Rightarrow \quad & \varepsilon_{j+n} = B^n \varepsilon_j. \end{aligned}$$

The propagation of errors in the numerical scheme is controlled by the matrix B. Let us fix a time, T, and compute the solution using a timestep of $h = T/m$. As $h \to 0$,

$$|\varepsilon_m| \sim |\lambda_1|^m |\varepsilon_1|,$$

where λ_1 is the largest eigenvalue of the matrix B given by Eq. (3.17). Clearly the FTCS scheme will be stable if $|\lambda_1| \leq 1$. Computing the eigenvalues of Eq. (3.17) is somewhat technical so I will just quote the answer:

$$\lambda_n = 1 - 4\delta \sin^2\left(\frac{n\pi}{2(N+1)}\right) \qquad n = 1, \ldots, N. \tag{3.33}$$

For stability we need $|\lambda_n| \leq 1$. $\delta > 0$ so we automatically have $\lambda_n \leq 1$. We need to ensure that $\lambda_n \geq -1$. Since $\sin^2(x) \leq 1$ it is sufficient to require $1 - 4\delta < -1$ which translates into

$$\delta = \frac{Dh}{(\Delta x)^2} < \frac{1}{2}. \tag{3.34}$$

We conclude that the FTCS scheme is conditionally stable. From this analysis and the Lax-Richtmyer Theorem quoted above, our numerical solution will converge to the analytical solution as $h \to 0$ and $\Delta x \to 0$ provided that we keep $h < \frac{(\Delta x)^2}{2D}$. From a theoretical perspective, this is exactly what we want. From a practical perspective, the fact that we have to decrease our timestep as the square of the spatial grid spacing in order to maintain stability as we increase the spatial resolution is a severe constraint on the efficiency of the method. We shall fix this problem in the next section.

3.3.3 Von Neumann Stability Analysis

This method of determining the stability of the FTCS scheme requires that we find the largest eigenvalue of the evolution matrix, a task which is mathematically difficult in general. We now introduce another approach to determinining numerical stability which, although less general, is easier in practice. The idea is to study the growth of a trial solution taking the form of a periodic wave.

Consider the FTCS scheme:

$$v_{n\,m+1} = v_{n\,m} + \delta\,(v_{n+1\,m} - 2v_{n\,m} + v_{n-1\,m}) \tag{3.35}$$

with a trial solution $v(x,t) = a(t)\,\mathrm{e}^{i\,k\,x}$. This trial solution leads to

$$a(t_{m+1})\mathrm{e}^{i\,k\,x_n} = a(t_m)\,\mathrm{e}^{i\,k\,x}\left[1 + \delta\left(\mathrm{e}^{i\,k\,\Delta x} - 2 + \mathrm{e}^{-i\,k\,\Delta x}\right)\right]. \tag{3.36}$$

It is then easy to show that

$$\frac{a(t_{m+1})}{a(t_m)} = 1 + \delta\,(2\cos(k\,\Delta x) - 2).$$

For stability we require

$$\left|\frac{a(t_{m+1})}{a(t_m)}\right| \leq 1$$

or alternatively

$$-1 \leq 1 - 2\delta\,(1 - \cos(k\,\Delta x)) \leq 1.$$

The latter inequality is clearly satisfied since $0 \leq 1 - \cos(x) \leq 2$. The first inequality requires that $\delta < \frac{1}{2}$. This is the same stability criterion which we obtained via the matrix method previously.

3.4 The Crank–Nicholson Method

The Crank–Nicholson method is an improvement on the FTCS scheme which is *unconditionally* stable. Of course there is a price to pay: the method is implicit. The idea is to base the finite difference scheme on the point $(x_i, t_j + \frac{h}{2})$. That is we approximate the equation:

$$\frac{\partial v}{\partial t}(x_i, t_j + \frac{h}{2}) = D\frac{\partial^2 v}{\partial x^2}(x_i, t_j + \frac{h}{2}). \tag{3.37}$$

We use a centred difference formula for the time derivative and approximate the spatial derivative by the average of the second order difference approximation at t_j and t_{j+1}:

$$\begin{aligned}
\frac{\partial v}{\partial t}(x_i, t_j + \frac{h}{2}) &= \frac{v_{i,j+1} - v_{i,j}}{h} + O(h^2) \\
\frac{\partial^2 v}{\partial x^2}(x_i, t_j + \frac{h}{2}) &= \frac{D}{2}\left[\frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{(\Delta x)^2} + \frac{v_{i+1,j+1} - 2v_{i,j+1} + v_{i-1,j+1}}{(\Delta x)^2}\right] + O((\Delta x)^2)
\end{aligned}$$

Note that the Crank–Nicholson scheme is also more accurate than the FTCS scheme. We thus arrive at the finite difference equation

$$\frac{v_{i,j+1} - v_{i,j}}{h} = \frac{D}{2}\left[\frac{v_{i+1,j} - 2v_{i,j} + v_{i-1,j}}{(\Delta x)^2} + \frac{v_{i+1,j+1} - 2v_{i,j+1} + v_{i-1,j+1}}{(\Delta x)^2}\right]. \tag{3.38}$$

We can rearrange this to give a set of equations relating the $v_{i,j+1}$ to the $v_{i,j}$:

$$-\frac{\delta}{2}v_{i+1,j+1} + (1+\delta)v_{i,j+1} - \frac{\delta}{2}v_{i-1,j+1} = \frac{\delta}{2}v_{i+1,j} + (1-\delta)v_{i,j} + \frac{\delta}{2}v_{i-1,j}, \tag{3.39}$$

where $\delta = \frac{Dh}{(\Delta x)^2}$. The method is implicit since we must solve a set of N simultaneous equations in order to obtain the $v_{i,j+1}$ from the $v_{i,j}$. What about the boundaries? Suppose we impose Dirichlet conditions on the spatial boundaries: $x = x_L$ and $x = x_R$:

$$\begin{aligned}
v(x_L, t) &= D_L(t) \\
v(x_R, t) &= D_R(t).
\end{aligned}$$

We introduce fictitious points, x_{-1} and x_N as we did in Sec. 3.2 and impose the boundary conditions on these points. At $i = 0$ we have the approximation:

$$\frac{v_{0,j+1} - v_{0,j}}{h} = \frac{D}{2}\left[\frac{v_{1,j} - 2v_{0,j} + D_L(t_j)}{(\Delta x)^2} + \frac{v_{1,j+1} - 2v_{0,j+1} + D_L(t_{j+1})}{(\Delta x)^2}\right]. \tag{3.40}$$

At $i = N - 1$ we have the approximation:

$$\frac{v_{N-1,j+1} - v_{N-1,j}}{h} = \frac{D}{2}\left[\frac{D_R(t_j) - 2v_{N-1,j} + v_{N-2,j}}{(\Delta x)^2} + \frac{D_R(t_{j+1}) - 2v_{N-1,j+1} + v_{N-2,j+1}}{(\Delta x)^2}\right]. \tag{3.41}$$

Eqs. (3.40) and (3.41) can be arranged to give the appropriate boundary equations to supplement Eq. (3.39):

$$\begin{aligned} -\frac{\delta}{2}v_{1,j+1} + (1+\delta)v_{0,j+1} &= \frac{\delta}{2}v_{1,j} + (1-\delta)v_{0,j} + \frac{\delta}{2}D_L(t_{j+1}) + \frac{\delta}{2}D_L(t_j) \\ (1+\delta)v_{N-1,j+1} - \frac{\delta}{2}v_{N-2,j+1} &= (1-\delta)v_{N-1,j} + \frac{\delta}{2}v_{N-2,j} + \frac{\delta}{2}D_R(t_{j+1}) + \frac{\delta}{2}D_R(t_j). \end{aligned} \tag{3.42}$$

After multiplying across by 2 for convenience, Eq. (3.39) together with Eqs, (3.42) can be concisely expressed as a linear system:

$$A\mathbf{v}_{j+1} = B\mathbf{v}_j + \mathbf{b}_{j+1} + \mathbf{b}_j, \tag{3.43}$$

where

$$A = \begin{pmatrix} 2(1+\delta) & -\delta & 0 & 0 & 0 \\ -\delta & 2(1+\delta) & -\delta & 0 & 0 \\ 0 & -\delta & 2(1+\delta) & -\delta & 0 \\ 0 & 0 & -\delta & 2(1+\delta) & -\delta \\ 0 & 0 & 0 & -\delta & 2(1+\delta) \end{pmatrix} \qquad \mathbf{b}_{j+1} = \begin{pmatrix} \delta D_L(t_{j+1}) \\ 0 \\ 0 \\ 0 \\ \delta D_R(t_{j+1}) \end{pmatrix}, \tag{3.44}$$

and

$$B = \begin{pmatrix} 2(1-\delta) & \delta & 0 & 0 & 0 \\ \delta & 2(1-\delta) & \delta & 0 & 0 \\ 0 & \delta & 2(1-\delta) & \delta & 0 \\ 0 & 0 & \delta & 2(1-\delta) & \delta \\ 0 & 0 & 0 & \delta & 2(1-\delta) \end{pmatrix} \qquad \mathbf{b}_j = \begin{pmatrix} \delta D_L(t_j) \\ 0 \\ 0 \\ 0 \\ \delta D_R(t_j) \end{pmatrix}, \tag{3.45}$$

We can solve the required set of equations at each step as follows:

$$\mathbf{v}_{j+1} = (A^{-1}\,B)\mathbf{v}_j + A^{-1}\mathbf{b}_{j+1} + A^{-1}\mathbf{b}_j. \tag{3.46}$$

In this example, we only need to invert the matrix A once at the beginning of the calculation since it does not change from one timestep to the next. In more complicated problems, for example if D were time-dependent or if adaptive stepping were used (in which case, δ would vary in time), then A would be different at each step. In these cases, a full matrix inversion is required at each step. This is potentially expensive but as mentioned already, fast algorithms exist for performing inversions of the kind of banded matrices which result from discretisation of differential operators like $\frac{\partial^2}{\partial x^2}$.

A similar set of steps can be followed to implement Neumann boundary conditions within the Crank–Nicholson scheme.

Fig. 3.2 shows some snapshots of the numerical solutions obtained with the Crank–Nicholson scheme with the same set of parameters for which the FTCS scheme was unstable (see Fig. 3.1). The left panel shows how the error varies as a function of δ for the two schemes. The plot confirms our analysis of the previous section that the FTCS scheme is conditionally stable (stable if $\delta < \frac{1}{2}$) and suggests that the Crank–Nicholson scheme is unconditionally stable. We now perform a mathematical analysis which confirms that this is the case.

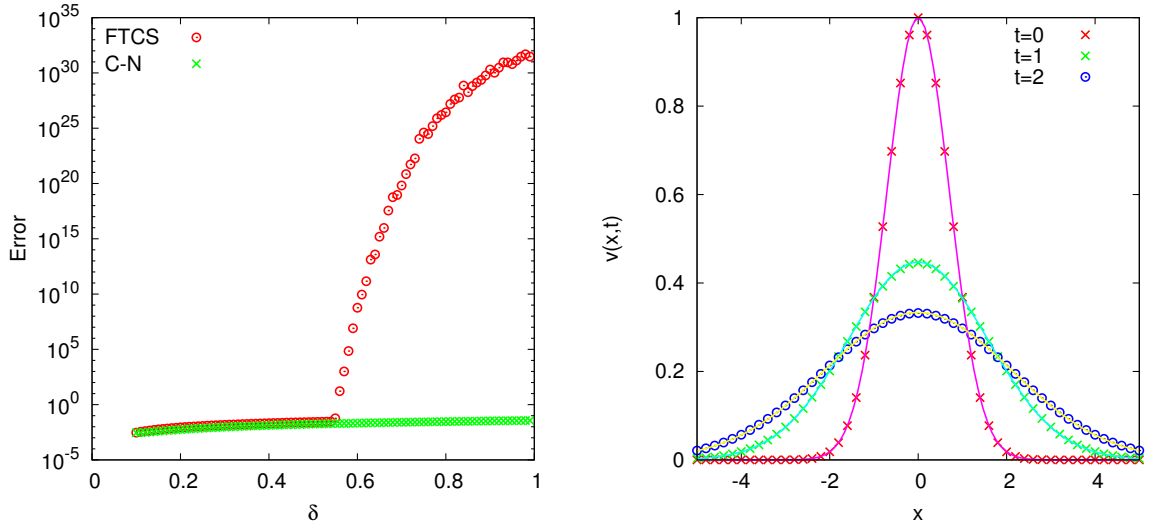

Figure 3.2: Left Panel: comparision of the numerical error made by the FTCS and Crank–Nicholson schemes as a function of δ for the problem described in Fig. 3.1. The conditional stability of the FTCS scheme is clearly evident whereas the Crank-Nicholson scheme is suggested to be unconditionally stable. Right Panel: explicit snapshots of the numerical results obtained using the Crank–Nicholson scheme for $\delta = 0.665$. Compare the corresponding results for the FTCS scheme in the right panel of Fig. 3.1.

3.5 Stability of the Crank–Nicholson Scheme

The consistency of the Crank–Nicholson scheme is a rather lenghty piece of analysis which does not differ significantly from that performed in Sec. 3.3 for the FTCS scheme. We omit it here.

The stability of the Crank–Nicholson scheme requires that we understand the structure of the matrix $A^{-1}\,B$ in Eq. (3.46)with A and B given by Eqs. (3.44) and (3.45). We just quote the result for the eigenvalues of $A^{-1}\,B$:

$$\lambda_n = \frac{2 - 4\delta \sin^2\left(\frac{n\pi}{2N}\right)}{2 + 4\delta \sin^2\left(\frac{n\pi}{2N}\right)}. \tag{3.47}$$

Note that for $\delta = 0$, $\lambda_n = 1$ for all values of n. As $\delta \to \infty$, $\lambda_n \to -1$ for all values of n. In general,

$$\begin{aligned} |\lambda_n| &= \frac{\left|2 - 4\delta \sin^2\left(\frac{n\pi}{2N}\right)\right|}{2 + 4\delta \sin^2\left(\frac{n\pi}{2N}\right)} \\ &\leq \frac{2 + 4\delta \sin^2\left(\frac{n\pi}{2N}\right)}{2 + 4\delta \sin^2\left(\frac{n\pi}{2N}\right)} \\ &= 1. \end{aligned}$$

Hence the Crank–Nicholson scheme is *unconditionally* stable.

3.6 Similarity solutions as attractors: rescaling

In our discussion of similarity solutions we mentioned briefly that, although such solutions are special, they are often attracting in an asymtotic sense.That is, lots of initial conditions converge to the similarity solution at large times provided that there are no boundary conditions or terms in the equation which break the scaling symmetry. In chap. 2 we lacked the technology to be able to verify this claim since we could only find the self-similar solution in isolation and could not match it to arbitrary initial conditions.

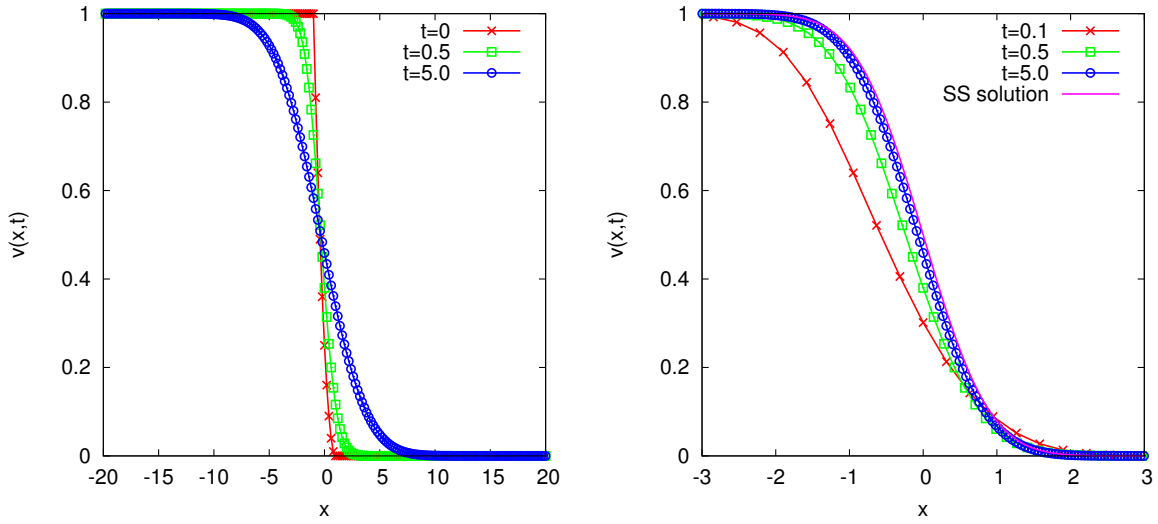

Figure 3.3: Snapshots of the numerical solution to Eq. (3.1) with the initial conditions given by Eq. (3.48) and the Dirichlet boundary conditions. Left panel shows the raw numerical solution and the right panel shows the numerical solution rescaled according to Eq. (3.49).

Now, however, equipped with the FTCS or Crank–Nicholson methods, we can solve the diffusion equation for arbitrary initial conditions and check if this is indeed the case. Let us reconsider the diffusing interface problem which we earlier described using the self-similar solution Eq. (2.47). Now suppose that our initial interface does not have a step function profile but something asymmetric:

$$v(x,0) = \begin{cases} 1 & x < -1 \\ \frac{1}{4}(1-x)^2 & -1 \leq x \leq 1 \\ 0 & x > 1. \end{cases} \tag{3.48}$$

This clearly does not "fit" with the self–similar $\mathrm{Erf}(x)$ profile. However that does not pose any problem to the methods which we have developed in the last few sections. If the self–similar solution is attracting however, we should find that at large times, our numerical solution, $\widetilde{v}(x,t)$, should behave as

$$\widetilde{v}(x,t) \approx t^a F(xt^b) \tag{3.49}$$

with a and b determined as before. Suppose that we plot $\widetilde{v}(x,t)/t^a$ as a function of xt^b then the data should collapse onto the single curve $F(x)$ as t gets large. This procedure is illustrated in Fig. 3.3. The numerical solution has been computed using the Crank–Nicholson method starting with the above initial data on the spatial intervale $[-20:20]$ with Dirichlet boundary conditions $v(-20,t) = 1$ and $v(20,t) = 0$. Note that we have set things up so that the solution can evolve for a relatively long time without feeling the boundaries. This gives the solution time to converge to the similarity solution before the boundaries start to break the scaling symmetry. The conclusion to be drawn from Fig. 3.3 is that the solutions quickly adopts the $\mathrm{Erf}(x)$ profile predicted by the similarity analysis as we claimed would be the case.