The correct way to difference Schrödinger's equation [1,2] is to use *Cayley's form* for the finite-difference representation of e^{-iHt}, which is second-order accurate *and* unitary:

$$e^{-iHt} \simeq \frac{1 - \frac{1}{2}iH\Delta t}{1 + \frac{1}{2}iH\Delta t} \tag{19.2.35}$$

In other words,

$$\left(1 + \tfrac{1}{2}iH\Delta t\right)\psi_j^{n+1} = \left(1 - \tfrac{1}{2}iH\Delta t\right)\psi_j^n \tag{19.2.36}$$

On replacing H by its finite-difference approximation in x, we have a complex tridiagonal system to solve. The method is stable, unitary, and second-order accurate in space and time. In fact, it is simply the Crank-Nicholson method once again!

CITED REFERENCES AND FURTHER READING:

Ames, W.F. 1977, *Numerical Methods for Partial Differential Equations*, 2nd ed. (New York: Academic Press), Chapter 2.

Goldberg, A., Schey, H.M., and Schwartz, J.L. 1967, *American Journal of Physics*, vol. 35, pp. 177–186. [1]

Galbraith, I., Ching, Y.S., and Abraham, E. 1984, *American Journal of Physics*, vol. 52, pp. 60–68. [2]

19.3 Initial Value Problems in Multidimensions

The methods described in §19.1 and §19.2 for problems in $1 + 1$ dimension (one space and one time dimension) can easily be generalized to $N + 1$ dimensions. However, the computing power necessary to solve the resulting equations is enormous. If you have solved a one-dimensional problem with 100 spatial grid points, solving the two-dimensional version with 100×100 mesh points requires *at least* 100 times as much computing. You generally have to be content with very modest spatial resolution in multidimensional problems.

Indulge us in offering a bit of advice about the development and testing of multidimensional PDE codes: You should always first run your programs on *very small* grids, e.g., 8×8, even though the resulting accuracy is so poor as to be useless. When your program is all debugged and demonstrably stable, *then* you can increase the grid size to a reasonable one and start looking at the results. We have actually heard someone protest, "my program would be unstable for a crude grid, but I am sure the instability will go away on a larger grid." That is nonsense of a most pernicious sort, evidencing total confusion between accuracy and stability. In fact, new instabilities sometimes do show up on *larger* grids; but old instabilities never (in our experience) just go away.

Forced to live with modest grid sizes, some people recommend going to higher-order methods in an attempt to improve accuracy. This is very dangerous. Unless the solution you are looking for is known to be smooth, and the high-order method you are using is known to be extremely stable, we do not recommend anything higher than second-order in time (for sets of first-order equations). For spatial differencing, we recommend the order of the underlying PDEs, perhaps allowing second-order spatial differencing for first-order-in-space PDEs. When you increase the order of a differencing method to greater than the order of the original PDEs, you introduce spurious solutions to the difference equations. This does not create a problem if they all happen to decay exponentially; otherwise you are going to see all hell break loose!

Lax Method for a Flux-Conservative Equation

As an example, we show how to generalize the Lax method (19.1.15) to two dimensions for the conservation equation

$$\frac{\partial u}{\partial t} = -\nabla \cdot \mathbf{F} = -\left(\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y}\right) \tag{19.3.1}$$

Use a spatial grid with

$$\begin{aligned} x_j &= x_0 + j\Delta \\ y_l &= y_0 + l\Delta \end{aligned} \tag{19.3.2}$$

We have chosen $\Delta x = \Delta y \equiv \Delta$ for simplicity. Then the Lax scheme is

$$\begin{aligned} u_{j,l}^{n+1} = &\frac{1}{4}(u_{j+1,l}^n + u_{j-1,l}^n + u_{j,l+1}^n + u_{j,l-1}^n) \\ &- \frac{\Delta t}{2\Delta}(F_{j+1,l}^n - F_{j-1,l}^n + F_{j,l+1}^n - F_{j,l-1}^n) \end{aligned} \tag{19.3.3}$$

Note that as an abbreviated notation F_{j+1} and F_{j-1} refer to F_x, while F_{l+1} and F_{l-1} refer to F_y.

Let us carry out a stability analysis for the model advective equation (analog of 19.1.6) with

$$F_x = v_x u, \qquad F_y = v_y u \tag{19.3.4}$$

This requires an eigenmode with two dimensions in space, though still only a simple dependence on powers of ξ in time,

$$u_{j,l}^n = \xi^n e^{ik_x j\Delta} e^{ik_y l\Delta} \tag{19.3.5}$$

Substituting in equation (19.3.3), we find

$$\xi = \frac{1}{2}(\cos k_x\Delta + \cos k_y\Delta) - i\alpha_x \sin k_x\Delta - i\alpha_y \sin k_y\Delta \tag{19.3.6}$$

where

$$\alpha_x = \frac{v_x\Delta t}{\Delta}, \qquad \alpha_y = \frac{v_y\Delta t}{\Delta} \tag{19.3.7}$$

The expression for $|\xi|^2$ can be manipulated into the form

$$\begin{aligned} |\xi|^2 = 1 - (\sin^2 k_x\Delta + \sin^2 k_y\Delta)\left[\frac{1}{2} - (\alpha_x^2 + \alpha_y^2)\right] \\ - \frac{1}{4}(\cos k_x\Delta - \cos k_y\Delta)^2 - (\alpha_y \sin k_x\Delta - \alpha_x \sin k_y\Delta)^2 \end{aligned} \tag{19.3.8}$$

The last two terms are negative, and so the stability requirement $|\xi|^2 \le 1$ becomes

$$\frac{1}{2} - (\alpha_x^2 + \alpha_y^2) \ge 0 \tag{19.3.9}$$

or

$$\Delta t \le \frac{\Delta}{\sqrt{2}(v_x^2 + v_y^2)^{1/2}} \tag{19.3.10}$$

This is an example of the general result for the N-dimensional Courant condition: If $|v|$ is the maximum propagation velocity in the problem, then

$$\Delta t \le \frac{\Delta}{\sqrt{N}|v|} \tag{19.3.11}$$

is the Courant condition.

Diffusion Equation in Multidimensions

Let us consider the two-dimensional diffusion equation,

$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right) \tag{19.3.12}$$

An explicit method, such as FTCS, can be generalized from the one-dimensional case in the obvious way. However, we have seen that diffusive problems are usually best treated implicitly. Suppose we try to implement the Crank-Nicholson scheme in two dimensions. This would give us

$$u_{j,l}^{n+1} = u_{j,l}^n + \frac{1}{2}\alpha\left(\delta_x^2 u_{j,l}^{n+1} + \delta_x^2 u_{j,l}^n + \delta_y^2 u_{j,l}^{n+1} + \delta_y^2 u_{j,l}^n\right) \tag{19.3.13}$$

Here

$$\alpha \equiv \frac{D\Delta t}{\Delta^2} \qquad \Delta \equiv \Delta x = \Delta y \tag{19.3.14}$$

$$\delta_x^2 u_{j,l}^n \equiv u_{j+1,l}^n - 2u_{j,l}^n + u_{j-1,l}^n \tag{19.3.15}$$

and similarly for $\delta_y^2 u_{j,l}^n$. This is certainly a viable scheme; the problem arises in solving the coupled linear equations. Whereas in one space dimension the system was tridiagonal, that is no longer true, though the matrix is still very sparse. One possibility is to use a suitable sparse matrix technique (see §2.7 and §19.0).

Another possibility, which we generally prefer, is a slightly different way of generalizing the Crank-Nicholson algorithm. It is still second-order accurate in time and space, and unconditionally stable, but the equations are easier to solve than

(19.3.13). Called the *alternating-direction implicit method (ADI)*, this embodies the powerful concept of *operator splitting* or *time splitting*, about which we will say more below. Here, the idea is to divide each timestep into two steps of size $\Delta t/2$. In each substep, a different dimension is treated implicitly:

$$\begin{aligned} u_{j,l}^{n+1/2} &= u_{j,l}^{n} + \frac{1}{2}\alpha\left(\delta_x^2 u_{j,l}^{n+1/2} + \delta_y^2 u_{j,l}^{n}\right) \\ u_{j,l}^{n+1} &= u_{j,l}^{n+1/2} + \frac{1}{2}\alpha\left(\delta_x^2 u_{j,l}^{n+1/2} + \delta_y^2 u_{j,l}^{n+1}\right) \end{aligned} \tag{19.3.16}$$

The advantage of this method is that each substep requires only the solution of a simple tridiagonal system.

Operator Splitting Methods Generally

The basic idea of operator splitting, which is also called *time splitting* or *the method of fractional steps*, is this: Suppose you have an initial value equation of the form

$$\frac{\partial u}{\partial t} = \mathcal{L}u \tag{19.3.17}$$

where $\mathcal{L}$ is some operator. While $\mathcal{L}$ is not necessarily linear, suppose that it can at least be written as a linear sum of m pieces, which act additively on u,

$$\mathcal{L}u = \mathcal{L}_1 u + \mathcal{L}_2 u + \cdots + \mathcal{L}_m u \tag{19.3.18}$$

Finally, suppose that for *each* of the pieces, you already know a differencing scheme for updating the variable u from timestep n to timestep $n+1$, valid if that piece of the operator were the *only* one on the right-hand side. We will write these updatings symbolically as

$$\begin{aligned} u^{n+1} &= \mathcal{U}_1(u^n, \Delta t) \\ u^{n+1} &= \mathcal{U}_2(u^n, \Delta t) \\ &\cdots \\ u^{n+1} &= \mathcal{U}_m(u^n, \Delta t) \end{aligned} \tag{19.3.19}$$

Now, one form of operator splitting would be to get from n to $n+1$ by the following sequence of updatings:

$$\begin{aligned} u^{n+(1/m)} &= \mathcal{U}_1(u^n, \Delta t) \\ u^{n+(2/m)} &= \mathcal{U}_2(u^{n+(1/m)}, \Delta t) \\ &\cdots \\ u^{n+1} &= \mathcal{U}_m(u^{n+(m-1)/m}, \Delta t) \end{aligned} \tag{19.3.20}$$

For example, a combined advective-diffusion equation, such as

$$\frac{\partial u}{\partial t} = -v\frac{\partial u}{\partial x} + D\frac{\partial^2 u}{\partial x^2} \tag{19.3.21}$$

might profitably use an explicit scheme for the advective term combined with a Crank-Nicholson or other implicit scheme for the diffusion term.

The alternating-direction implicit (ADI) method, equation (19.3.16), is an example of operator splitting with a slightly different twist. Let us reinterpret (19.3.19) to have a different meaning: Let $\mathcal{U}_1$ now denote an updating method that includes algebraically *all* the pieces of the total operator $\mathcal{L}$, but which is desirably *stable* only for the $\mathcal{L}_1$ piece; likewise $\mathcal{U}_2, \ldots \mathcal{U}_m$. Then a method of getting from u^n to u^{n+1} is

$$\begin{aligned} u^{n+1/m} &= \mathcal{U}_1(u^n, \Delta t/m) \\ u^{n+2/m} &= \mathcal{U}_2(u^{n+1/m}, \Delta t/m) \\ &\cdots \\ u^{n+1} &= \mathcal{U}_m(u^{n+(m-1)/m}, \Delta t/m) \end{aligned} \tag{19.3.22}$$

The timestep for each fractional step in (19.3.22) is now only $1/m$ of the full timestep, because each partial operation acts with all the terms of the original operator.

Equation (19.3.22) is usually, though not always, stable as a differencing scheme for the operator $\mathcal{L}$. In fact, as a rule of thumb, it is often sufficient to have stable $\mathcal{U}_i$'s only for the operator pieces having the highest number of spatial derivatives — the other $\mathcal{U}_i$'s can be *unstable* — to make the overall scheme stable!

It is at this point that we turn our attention from initial value problems to boundary value problems. These will occupy us for the remainder of the chapter.

CITED REFERENCES AND FURTHER READING:

Ames, W.F. 1977, *Numerical Methods for Partial Differential Equations*, 2nd ed. (New York: Academic Press).

19.4 Fourier and Cyclic Reduction Methods for Boundary Value Problems

As discussed in §19.0, most boundary value problems (elliptic equations, for example) reduce to solving large sparse linear systems of the form

$$\mathbf{A} \cdot \mathbf{u} = \mathbf{b} \tag{19.4.1}$$

either once, for boundary value equations that are linear, or iteratively, for boundary value equations that are nonlinear.