The Conception, Gestation, Birth, and Infancy of FCT

David L. Book

Abstract How Flux-Corrected Transport came to be, as recalled by one of the innovators: recollections of how FCT was developed and of the individuals responsible.

1 Conception

In 1970 there was essentially no reliable way to solve fluid equations numerically. By 1971 there was one. Flux-Corrected Transport (FCT) was the first method developed that yielded physically acceptable results for such equations. The present paper describes how Jay Boris and I developed FCT, with what I hope is an accurate account of our thinking at the time, the path we followed in the course of the development, including the missteps and blind alleys, and the roles of the other individuals who were involved.

Fluid or hydrodynamic equations are partial differential equations dominated by convective motion, that is, equations in which convective derivative terms of the form

$$\frac{\partial f}{\partial t} + \mathbf{v} \cdot \nabla f$$

play a decisive role, where f is one of the dependent fluid variables (density of mass, momentum, energy, or charge; pressure, entropy, species concentration, etc.), t is time, $\mathbf{v}$ is the flow velocity, $\nabla \equiv \frac{\partial}{\partial \mathbf{r}}$, and $\mathbf{r}$ is position. Examples are the continuity equation for a compressible medium with mass density ρ,

$$\frac{\partial \rho}{\partial t} + \nabla \cdot \rho \mathbf{v} = 0,$$

the Euler (momentum) equation in the presence of a scalar pressure p and a constant gravitational acceleration $\mathbf{g}$, which can be written

$$\frac{\partial (\rho \mathbf{v})}{\partial t} + \nabla \cdot \rho \mathbf{v}\mathbf{v} + \nabla p + \rho \mathbf{g} = 0,$$

D.L. Book (✉)
Enigmatics, Inc., P.O. Box 8610, Monterey, CA 93943, USA
e-mail: davidbook@enigmatics.com

D. Kuzmin et al. (eds.), *Flux-Corrected Transport*, Scientific Computation,
DOI 10.1007/978-94-007-4038-9_1,

and the Navier–Stokes equation (the Euler equation with the inclusion of viscosity),

$$\rho \frac{\partial \mathbf{v}}{\partial t} + \rho \mathbf{v} \cdot \nabla \mathbf{v} + \nabla p = \mu \nabla^2 \mathbf{v}.$$

Such equations are also called convective or hyperbolic, although strictly speaking the Navier–Stokes equation is parabolic in regions where the Laplacian term dominates. In general, the equation expressing the transport of any continuously distributed quantity, together with terms describing sources and losses, is of this form. This category includes all of the familiar conservation laws.

A third of a century ago computer resources were meager by comparison with today's technology, but most of the general computational approaches in use today were known, at least in broad outline, and the first steps had already been taken toward applying them. Finite differences and finite elements (the distinction between them then was somewhat blurred, though now people usually associate these terms with differencing schemes on structured and unstructured grids, respectively), characteristics, quasiparticles, and spectral methods had all been invented, and all of these tools were being applied to the problem of finding computational solutions to fluid equations. Dozens of Fortran codes based on each of these methods, or combinations of several of them, were in existence. They all fell short of what I feel is the goal of any numerical treatment of evolution equations: using limited, i.e., discretized, information about the dependent variables in order to predict the values of those variables at a later time *with the same accuracy or level of confidence*.

The presence of convective derivatives is what makes fluid equations difficult to solve numerically. Because of them the characteristics, the space–time trajectories along which the values of the fluid variables are constant, slope. In order to predict the values at a position $\mathbf{r}$ of the fluid variables at a time t' later than a time t for which their values are known, it is necessary to use information from the points through which the characteristics passed at time t, which are in general different from $\mathbf{r}$. Thus, to predict the values of the fluid variables at a particular point on, e.g., a finite-difference grid may require knowing their values at an earlier time from a location *that was not on the grid*.

In thinking about the implications of sloping characteristics I like to use an analogy with what I call the window problem. Suppose you are in a room with several windows looking out onto a nearby railroad track. When a train comes by each car in succession appears at a given window. Let's say that the train is moving from your left to your right. If you want to know what car is going to appear next at the window in front of you, you can look out the window to the left of it. You don't have to keep a continuous watch; it suffices to glance over at the second window at intervals, which correspond to the discrete timesteps in a finite-difference scheme. Of course, if the windows are too widely spaced, or if you are too close to them, there may be several cars hidden out of sight between the two windows. This is analogous to using too coarse a mesh in your difference scheme. Likewise, you may miss a car if the intervals between glances are too long, which corresponds to using too long a timestep Δt.

But the information you get is limited in another way, because the window is narrow and you can't see a whole railroad car at one time. In order to make good

Fig. 1 Pathological railcars

predictions you have to know something about the kinds of railroad cars that the train is allowed to have. A glimpse may be all you need to know that a car is a boxcar or a tank car or a flat car, but what if it's a car of a type you've never seen before, say, one carrying a crane to lift up wrecks? Or one with unusual dimensions or proportions (Fig. 1)? Your prediction is implicitly based on a bias or prejudice in favor of the kinds of cars you expect to see.

In the same way, a numerical technique must contain a built-in bias about the form solutions can take, because the available information is limited by discretization. No technique can handle all situations. Each one must be tailored to fit a particular class of problems.

The problems Jay and I were interested in were time-dependent fluid problems, especially those involving supersonic flow, and more generally, systems with discontinuities or steep gradients. These include not just shocks (which can occur only when there is supersonic flow), but also contact and tangential discontinuities and abrupt changes in temperature, species concentration, etc. We opted to employ a finite-difference approach because it simplified coding, particularly in multidimensions and at boundaries, and because that was what we were most comfortable with.

The naïve approach to finding a finite-difference approximation to differential equations on a mesh with a uniform spacing Δx is to expand the derivatives in Taylor series:

$$f(x \pm \Delta x) = f(x) \pm \frac{\partial f}{\partial x}\Delta x + \frac{1}{2}\frac{\partial^2 f}{\partial x^2}(\Delta x)^2 \pm \cdots.$$

Thus,

$$f(x + \Delta x) - f(x - \Delta x) = 2\frac{\partial f}{\partial x}\Delta x + O(\Delta x)^3$$

or

$$\frac{\partial f(x,t)}{\partial x} = \frac{1}{2\Delta x}\big[f(x + \Delta x, t) - f(x - \Delta x, t)\big] + O(\Delta x)^2,$$

and similarly

$$\frac{\partial f(x,t)}{\partial t} = \frac{1}{2\Delta t}\big[f(x, t + \Delta t) - f(x, t - \Delta t)\big] + O(\Delta t)^2.$$

Hence a straightforward finite-difference approximation to the 1-D passive advection equation

$$\frac{\partial \rho}{\partial t} + u\frac{\partial \rho}{\partial x} = 0$$

is

$$\rho(x, t + \Delta t) = \rho(x, t - \Delta t) - \varepsilon\big[\rho(x + \Delta x, t) - \rho(x - \Delta x, t)\big],$$

where $\varepsilon = u\Delta t/\Delta x$ is the Courant number. The resulting finite-difference approximation is of course just second-order leapfrog, a useful scheme that has been widely adopted.

The Taylor-series approach has a number of strengths. It yields difference schemes (such as leapfrog) that are accurate for problems with slowly varying profiles, i.e., those in which discontinuities are absent. Also, it facilitates the analysis of amplitude and phase errors. Thus, assume a sinusoidal density profile

$$\rho_j^0 = \rho_0 \exp(2\pi i j\kappa/N),$$

where $j\Delta x$ is position, κ is the mode number and N is the number of mesh points. If

$$\rho_j^1 = \rho_1 \exp(2\pi i j\kappa/N)$$

is the corresponding profile found after one timestep, then the amplification factor is $A_\kappa = |\rho_1/\rho_0|$ and the relative phase error (the error in the speed with which features are advected numerically, divided by the correct speed) is $R_\kappa = (\kappa\varepsilon)^{-1}\tan^{-1}(\mathrm{Im}\,\rho_1/\mathrm{Re}\,\rho_0) - 1$.

But the Taylor-series expansion approach also has some notable deficiencies. It works only when "order" makes sense, i.e., when the scale of variation is large compared with the mesh spacing, so that the neglect of higher-order terms ("truncation errors") is justified. Consequently, it breaks down at discontinuities, where "dispersive" ripples make their appearance. (The finite-interval Gibbs effect, the analog for discrete Fourier transforms of the well-known Gibbs phenomenon, can also contribute to errors in the vicinity of a discontinuity. I will return to this topic below.)

The key insight (which to the best of my knowledge originated with Jay) is that the Taylor-series approach fails because it does not enforce *positivity*, a property sometimes called monotonicity. For physical reasons some variables can only take on positive values. Examples are mass and energy density (but not charge or momentum density), temperature, and pressure. Nothing in the Taylor-series approach ensures this. Positivity violations are found to be worst near discontinuities. At discontinuities formal high-order accuracy is less important than maintaining positivity.

This may be an appropriate time to try to spell out what is meant by the term "discontinuity." Shocks, contact discontinuities, and slip lines (tangential disconti-

nuities) would be physically discontinuous in the absence of dissipation. Because dissipation can never be totally absent, however, no physical quantity is ever truly discontinuous, at least in classical physics. There are no discontinuities in nature.

In finite-difference approximations, on the other hand, all changes are discontinuous, but some are more discontinuous than others. A criterion is needed to determine when a discontinuity is "real." This criterion may involve calculating whether the relative or absolute change in a variable exceeds some threshold value, or it may be more complicated. The exact definition, as always, depends on the nature of the problem and one's expectations about the solution being sought.

In the absence of a specific criterion a finite-difference scheme will not be able to distinguish a weak physical discontinuity from a smooth feature or from noise. As I will show, there are several reasons why a profile that should be smoothly varying might develop ripples or "bumps" in a computational treatment. Hence in any scheme susceptible to such errors nonphysical features will be treated just like physical ones. The trick is to avoid, as much as possible, developing them in the first place.

The simplest system of equations that models shocks is that of ideal hydrodynamics. This consists of the continuity and Euler equations and an equation for the energy density

$$E = \frac{1}{2}\rho \mathbf{v}^2 + \frac{p}{\gamma - 1}.$$

The energy equation follows from the adiabatic law (pressure equation) in the form

$$\frac{\partial p}{\partial t} + \mathbf{v} \cdot \nabla p + \gamma p \nabla \cdot \mathbf{v} = 0,$$

where γ is the ratio of specific heats.

These three equations express the conservation of mass, momentum, and energy. That they are conservative is important; if the adiabatic law is used instead of the formally equivalent energy equation, then pressure remains positive but the Rankine–Hugoniot (jump) conditions are violated. If the energy equation is chosen as one of the fundamental evolution equations, then energy is conserved and shocks obey the Rankine–Hugoniot conditions, but pressure is a derived quantity which can become negative. The Rankine–Hugoniot conditions imply that the jump in entropy varies as $(\mathrm{M} - 1)^3$, where M is the Mach number. Using the adiabatic law means that only weak shocks ($\mathrm{M} \to 1$) are calculated correctly. This dilemma confronts all finite-difference schemes.

Physically, viscous dissipation is responsible for creating entropy at a shock front. This is what allows shocks to satisfy the Rankine–Hugoniot conditions. The equations of ideal hydrodynamics in conservative form contain no explicit viscosity terms, but nevertheless admit solutions that satisfy the Rankine–Hugoniot conditions. These so-called weak solutions represent a zero-viscosity limit. Physical shocks in systems with nonzero viscosity differ in having nonzero thickness; that is, the jump takes place over a region of finite extent.

Any numerical scheme must include *some* dissipation if it is going to allow the jump conditions to be satisfied. Before the invention of FCT the most successful finite-difference treatments of supersonic flow, developed at Los Alamos and widely employed elsewhere, introduced an artificial viscosity term, typically in the form of a velocity-dependent coefficient multiplying the second difference of the velocity. This forced the production of entropy at places where the velocity underwent abrupt change and allowed the Rankine–Hugoniot conditions to hold.

The trouble with this approach was that in order to generate enough entropy the coefficient had to be large. This in turn caused the shock to be spread out over several or many mesh spaces. That was fine if the physical viscosity in the problem was large, i.e., if the Reynolds number *Re* was of order unity. But for realistic problems of high-Reynolds-number flow—say, $Re > 100$—it was hopelessly inaccurate. To reduce the shock thickness to reasonable values would require being able to determine where shocks were located or were about to form or introducing thousands of grid points in each coordinate direction, which would have been prohibitively expensive. (Remember, this was in 1970.)

There is another downside to using artificial viscosity. The timestep limit in finite-difference problems arises because information can travel no faster than one mesh space per timestep (assuming three-point difference schemes), which means that the Courant number must not exceed unity. Violating this condition in explicit finite-difference treatments of the ideal hydrodynamic equations leads to catastrophic numerical instability. If the spatial mesh in a calculation is refined the timestep must shrink proportionately to stay within safe limits. But when diffusion terms of the form $\mu\nabla^2\mathbf{v}$ dominate, then the timestep scales with the square of the mesh space. Consequently, in methods using artificial viscosity even refining the mesh locally near the shocks becomes much more expensive if the diffusion terms are differenced explicitly. (Implicit differencing has problems of its own.)

A shock wave heats the medium through which it is traveling, which causes the speed of sound to increase. Hence information propagates faster, so that signals in the region behind a shock tend to catch up with the shock. This means that shocks are self-steepening. Other discontinuities, such as contact surfaces, are not. As a result, shear surfaces and interfaces between two different media or between two regions in the same medium with different properties tend to be smeared out by numerical diffusion and are more difficult to model than shocks.

The passive advection equation models advection and propagates contact discontinuities, but in contrast to the system of hydrodynamic equations it has no self-steepening mechanism. Consequently, it is a more stringent test of numerical fluid-equation solvers. In addition it is simpler to work with than a set of nonlinear equations. Thus, it was natural for us to choose it as our test bench instead of the full set of hydrodynamic equations. We felt that if we could do a good job solving this equation numerically we could solve most fluid problems.

For our basic test problem we chose to propagate a square wave 20 mesh spaces in width across a grid 100 mesh spaces wide (Fig. 2) for 800 timesteps with a constant Courant number $\varepsilon = 0.2$, using periodic boundary conditions in order to al-

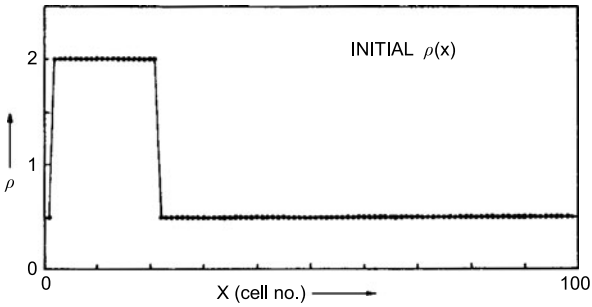

Fig. 2 Initial conditions for the square-wave test

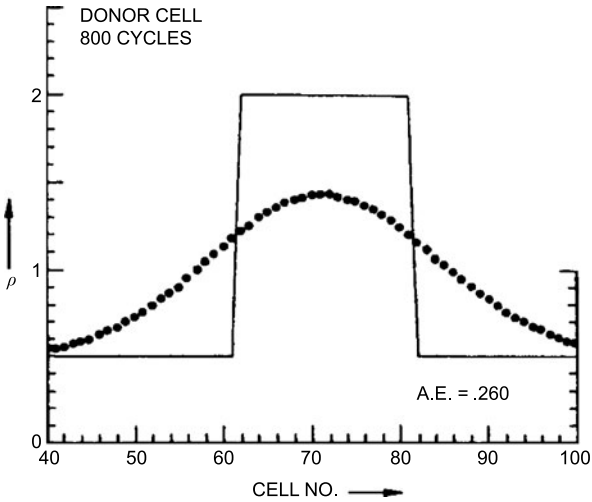

Fig. 3 Square-wave test of donor cell (first-order accuracy)

low the profile to reenter the system. As a quantitative figure of merit we used the average absolute value of the error (the L1 norm), abbreviated A.E. This test problem subsequently became a kind of universal standard in the computational physics community.

When various numerical methods are evaluated using this problem one can readily discern their strengths and shortcomings. Those that are inaccurate to zeroth order in terms of Taylor-series expansions in $k\Delta x$, the nondimensionalized wavenumber, fail to track the analytical solution correctly, yielding profiles that move either too fast or too slow. Those that are first-order accurate, such as donor-cell (upwind differencing), yield profiles that move at the right speed and maintain positivity, i.e., $\rho(x) > 0$ everywhere, but become smeared out over an ever-increasing portion of the grid (Fig. 3). In other words, they are highly *diffusive*.

Methods that are second-order accurate, such as leapfrog or Lax–Wendroff, yield profiles that develop multiple ripples (Fig. 4). These arise because the various Fourier harmonics that make up the square wave propagate at different speeds. The long-wavelength components propagate at nearly the right speed, while the short-wavelengths usually lag behind. In other words, the errors are *dispersive*. The rip-

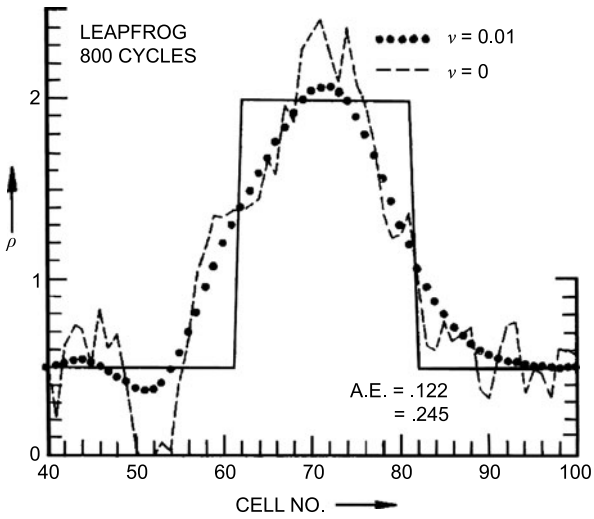

Fig. 4 Square-wave test of leapfrog (second-order accuracy)

ples grow in amplitude until the profile can become negative in some places. Thus, second-order algorithms do not maintain positivity. Notice that introducing a small amount of smoothing ($\nu = 0.01$) not only eliminates these negative values but also reduces the A.E. The optimum choice of the smoothing coefficient ν is, however, problem-dependent.

It is difficult to say which is worse, diffusive or dispersive errors, Scylla or Charybdis. Going to higher than second order doesn't solve the problem. Every technique that can be expressed in terms of linear finite-difference operations on the dependent variable—including every finite-difference treatment of the passive advection equation and its hydrodynamic kin in existence prior to the invention of FCT—suffers from one or the other failing.

2 Gestation

FCT was the first *nonlinear* finite-difference technique. In my view there were three main steps in our thinking that led to its development: expressing all operations in terms of fluxes, certainly not a new idea at the time; a transport algorithm called SHASTA, which is highly diffusive even in the limit of zero velocity, suggesting the use of "antidiffusion" to cancel out the diffusive errors; and the idea of correcting (limiting) the antidiffusive fluxes in order to maintain positivity (the nonlinear ingredient).

Fluxes are quantities of an extensive variable (e.g., mass, momentum, energy) that pass from one cell or grid point to another. If a finite-difference algorithm can be expressed entirely in terms of fluxes then it is guaranteed to be conservative, because

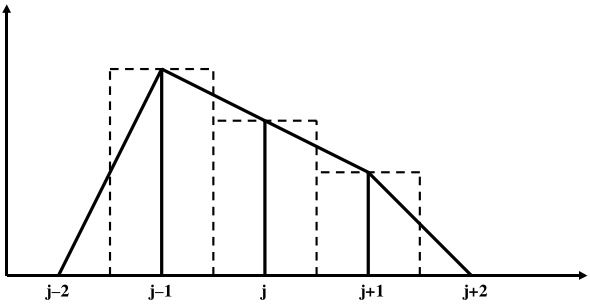

Fig. 5 Finite-difference approximation represented by rectangles and by trapezoids

what is removed from one place reappears in another. Thus, advection (transport only) can be approximated using transportative fluxes:

$$\rho_j^T = \rho_j - \phi_{j+1/2}^T + \phi_{j-1/2}^T,$$

where

$$\phi_{j+1/2}^T = \varepsilon_{j+1/2}\rho_{j+1/2},$$

with $\varepsilon_{j+1/2} = u_{j+1/2}\Delta t/\Delta x_{j+1/2}$ and $\rho_{j+1/2}$ defined on the cell boundary, $x_{j+1/2} = \frac{1}{2}(x_j + x_{j+1})$, and $\Delta x_{j+1/2} = x_{j+1} - x_j$.

Similarly, diffusion can be expressed in terms of diffusive fluxes:

$$\rho_j^D = \rho_j + \phi_{j+1/2}^D - \phi_{j-1/2}^D,$$

where

$$\phi_{j+1/2}^D = \nu_{j+1/2}(\rho_{j+1} - \rho_j).$$

In both instances *what is subtracted from one cell is added to its neighbor*, so the total "mass" is conserved.

The second ingredient, SHASTA, was Jay's idea. He devised it by means of a geometric approach. Imagine a finite-difference approximation ρ_j to some continuous variable $\rho(x)$, represented by histograms or rectangles (the broken lines in Fig. 5). Connect the points denoting the values of ρ_j with straight lines to form trapezoids. The area contained in the resulting trapezoids is the same as that contained in the rectangles.

If this profile is then transported across the grid with some velocity $u(x)$, in general each trapezoidal packet of fluid undergoes advection together with compression or expansion. Each mesh point x_j moves to a new location x'_j. In one timestep the two vertical sides x_j and x_{j+1} of a trapezoid thus move in general by different distances, causing it to be deformed as well as translated. The condition $x'_j < x'_{j+1}$, which is necessary to ensure positivity, imposes the limitation $\varepsilon < 0.5$. At the end of each timestep the mass contained in the trapezoid is reassigned to the two rectangles it straddles: the portion to the left of the boundary between two cells is assigned to the left-hand cell and the portion to the right is assigned to the right-hand cell (Fig. 6).

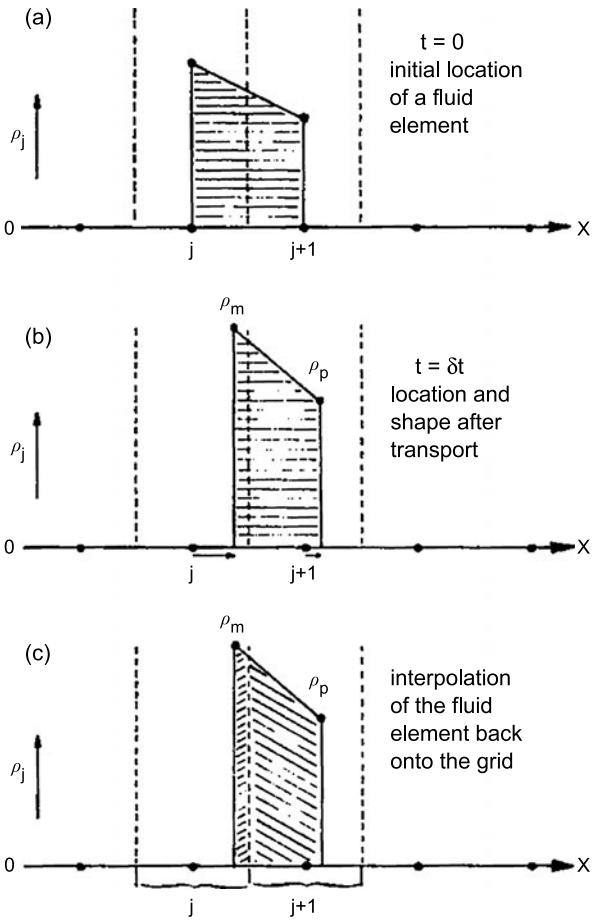

Fig. 6 SHASTA contains a zeroth-order diffusion

It is easy to see that the process of creating trapezoids and reassigning their mass is highly diffusive. In fact, in the limit when the velocity is identically zero the new value of ρ_j equals the old value plus a second difference with coefficient of 0.125:

$$\rho_j^{n+1} = \rho_j^n + \frac{1}{8}\left(\rho_{j+1}^n - 2\rho_j^n - \rho_{j-1}^n\right).$$

Since this algorithm is diffusive, the natural thing to do is to subtract the excess diffusion, or to put it another way, to apply antidiffusion. Antidiffusion is just diffusion with a negative coefficient:

$$\frac{\partial \rho}{\partial t} = -A\frac{\partial^2 \rho}{\partial x^2}, \quad A > 0.$$

Whereas diffusion erodes features, antidiffusion steepens them. Discontinuities become sharper and new extrema can occur. If this process is sufficiently drastic it can violate positivity (Fig. 7).

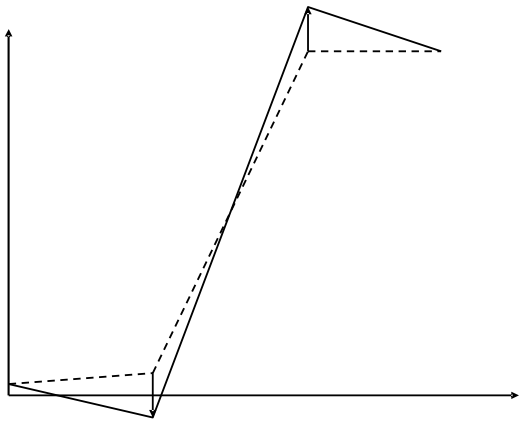

Fig. 7 Antidiffusion can violate positivity

One way to look at this is to recognize that diffusion converts a second-order algorithm like Lax–Wendroff or leapfrog that has wiggles into one that is first-order like donor cell. Hence taking out the excess diffusion changes it back into a second-order algorithm and restores the wiggles.

Jay recognized that by correcting or *limiting* the antidiffusive fluxes before they are applied he could avoid restoring the wiggles. A flux large enough to push a point j down below its neighbors will create a new minimum that might go negative, so it is necessary to reduce the size of this flux. If the profile already has a minimum at j, then allowing it to be pushed further down is dangerous, so it is necessary to zero any flux that tends to do so. Likewise, it is necessary to avoid enhancing maxima on negative profiles. The two rules can be combined into one: replace the "raw" antidiffusive fluxes with fluxes "corrected" so that *no new extrema can form and existing extrema cannot grow*. Figure 8 illustrates the four different situations a limiter can face when dealing with the flux between the points j and $j+1$, assuming the gradient is positive there: (a) no pre-existing extrema; (b) a maximum present at $j+1$; (c) a minimum present at j; or (d) both.

This was the first flux limiter that gave satisfactory results. We eventually realized that there are other workable variants, but this one, in which each flux is corrected without reference to others, is arguably the simplest. Because its action sometimes amounts to overkill (in ways I will describe shortly) I called it "strong" flux limiting.

The idea of using a flux limiter was the crucial ingredient in FCT. Some of the credit for it should go to the late Klaus Hain, our colleague at the Naval Research Laboratory. At the time Klaus was also trying to develop an algorithm to solve convective equations. I believe he was the first to recognize that some sort of adjustment or correction in the fluxes was needed, but he had not yet found the right formulation. Jay picked up the idea from him and made it work.

In some ways we were Klaus's competitors more than his collaborators. Klaus was an extremely able numericist, and Jay clearly wanted to be the first to find a successful algorithm. Jay and I worked closely together, but Klaus worked almost entirely alone. I thought it was because Klaus, German-born and about two decades

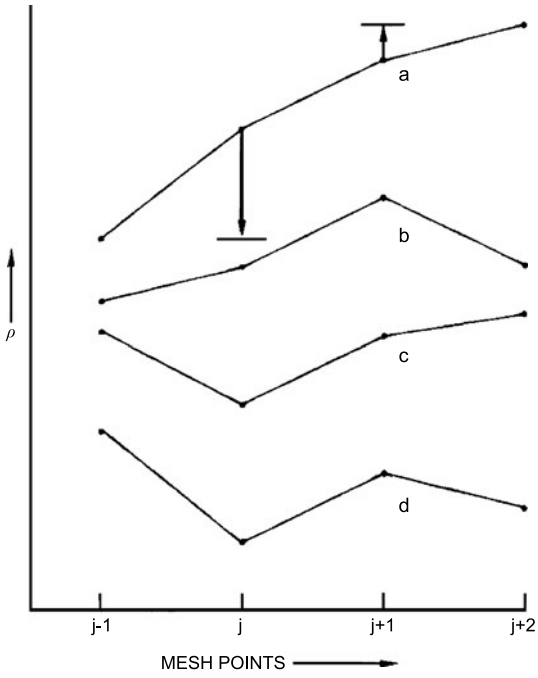

Fig. 8 Possible actions of a "strong" flux limiter on a positive flux

older than us, had trouble speaking and writing English. His wife Gertie, however, assured everyone that he was uncommunicative in German too.

3 Birth

By mid-1971 we had a working one-dimensional flux-corrected version of SHASTA. The name SHASTA supposedly came not from the famous mountain, but from the *nom de guerre* of a topless dancer. (I cannot confirm this from my own experience.) Possibly because he thought this lacking in dignity Jay contrived the acronym "SHarp And Smooth Transport Algorithm," which is how we presented it to the world.

Initially Jay had been unwilling to reveal the secret of FCT to outsiders, but one day he said to me, "Here comes Super-Klaus!" He suspected that Klaus was writing up his work for publication, and that was what changed his mind. We decided that one paper was not enough. Instead there would be a series, which became the celebrated FCT-1, FCT-2, and FCT-3. Jay was the obvious choice as the senior author of the first paper, most of which he drafted, but he must have felt a little guilty about scooping Klaus, because he suggested that the subsequent papers should bear the names of all three of us. Each of us would write the first draft of one of them; Klaus would be lead author of one and I would be lead author of the other. In the event though, Klaus—true to form—never did write anything, and his name appeared only on the second paper.

Actually, the first publications describing FCT appeared not in 1973, but two years earlier. (Hence the workshop for which I prepared the present paper should properly be called FCT-32, which is a more computational-sounding name than FCT-30 anyway.) I wrote the first journal article, which closely followed the manuscript we were preparing for submission to the Journal of Computational Physics at the time. It appeared in the November issue of NRL Reports of Progress, a house organ read by almost nobody. The only reason I wrote it was because I was then serving on a committee set up in order to make the Reports of Progress more relevant and better known.

But our very first publication was oral and never found its way into print. The application for which FCT was intended was modeling the atmospheric nuclear explosions that would have resulted from the infamous antiballistic missile program. The 1971 Symposium on High-Altitude Nuclear Effects (HANE), held at Stanford in August, was the first exposure of FCT to the computational community at large. The meeting was attended by government and private contractors funded by the Defense Atomic Support Agency, which that year became the Defense Nuclear Agency (DNA), afterward called the Defense Special Weapons Agency, and currently the Defense Threat Reduction Agency.

Several people from NRL were there, presenting results on various aspects of HANE. The attendees from other organizations were of course competing with us for DNA support. We wanted to impress them and our sponsor, but not to tell them too much. I gave the talk on the design of FCT. Jay was very insistent that I not reveal any secrets, and I didn't. I showed the results of the square-wave tests, I described SHASTA, but I didn't explain how the flux limiter worked, nor did I mention an embellishment called a "steepener." A lot of what I said was sheer doubletalk. I have a vivid recollection of a frustrated Greg Canavan from the Air Force Weapons Laboratory standing in the audience, asking question after question, trying to pin me down. Finally he said, "You keep saying, 'Another way to look at it is so-and-so.' Just tell us how it works!"

The meeting was a triumph for FCT and for our group at NRL. Our co-authors on that initial publication, Carl Wagner and Ed McDonald, were among the first users of SHASTA. Carl later worked on controlled fusion in private industry; Ed, who is still at NRL, changed fields and is now widely known for his work modeling sound propagation in the ocean. Jay and I both began applying SHASTA to a variety of problems, as did at least half a dozen of our colleagues in the Plasma Dynamics Branch. Many of these led to plasma and ionospheric physics papers.

At the same time we continued to refine and extend the method while preparing for publication in JCP. Some of the improvements resulted in making the code more efficient. Jay found that the flux limiter, which originally required a nested sequence of IF statements, could be expressed by a one-line formula:

$$\phi^c_{j+1/2} = \sigma_{j+1/2} \max\bigl[0, \min(\sigma_{j+1/2}\Delta_{j-1/2}, |\phi_{j+1/2}|, \sigma_{j+1/2}\Delta_{j+3/2})\bigr].$$

Here

$$\Delta_{j+1/2} = \rho^{TD}_{j+1} - \rho^{TD}_j,$$
$$\phi_{j+1/2} = \mu\bigl(\rho^{TD}_{j+1} - \rho^{TD}_j\bigr) \quad \text{or} \quad \phi_{j+1/2} = \mu\bigl(\rho^{T}_{j+1} - \rho^{T}_j\bigr),$$

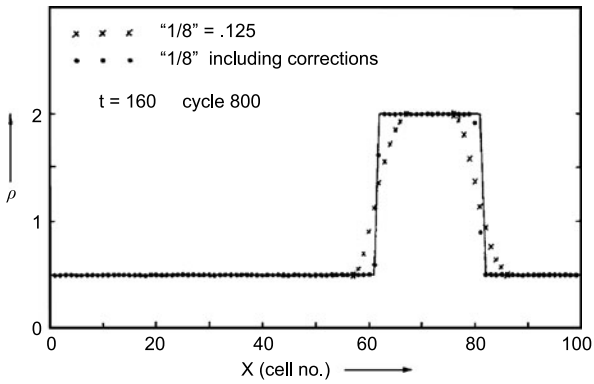

Fig. 9 The effect of introducing an artificial steepener

where ρ_j^T is the transported density, ρ_j^{TD} is the transported diffused density, and

$$\sigma_{j+1/2} = \text{sign}(\phi_{j+1/2}).$$

The two versions given here for the raw antidiffusive flux both use the same dimensionless coefficient μ, but the second version, which we called phoenical, has the advantage that when the flow velocity u vanishes $\rho_j^T \to \rho_j$. This permits the profile, which has been smeared out by diffusion, to be restored "like a phoenix," so that the algorithm reduces to the identity operation.

There was one aspect of the early versions of FCT that I felt uneasy about, the use of steepeners. In FCT-1 the antidiffusion coefficient μ was given as "1/8." We wrote "The quotation marks indicate that more exact cancellation of errors can be achieved if one expends a small amount of computational effort by including at least rough approximations to the velocity- and wavenumber-dependent corrections [11]." Footnote 11 explained just what was meant by wavenumber dependence: The antidiffusion coefficient was bigger than the diffusion coefficient by an amount that depended on the size of the discontinuity. Naturally, this yielded very nice square waves (Fig. 9). The drawback was that it turned every bump into a square wave!

The steepener was Jay's idea. The rationale for it was that our tests produced profiles that were actually less sharp than shocks calculated with FCT because, as I mentioned earlier, the passive advection equation has no mechanism for self-steepening. The steepener was supposed to model this mechanism, but to me it seemed an out-and-out kludge, and people who read footnote 11 apparently agreed. In the event steepeners disappeared after FCT-1 and were never mentioned again.

4 Infancy

As we continued to improve and extend SHASTA we gradually realized that what we had was more widely applicable and more general than a mere algorithm. Jay insisted on using the name "flux-corrected transport," but it was several years before the rest of the community distinguished between FCT and SHASTA. (Jay also tried

to reserve the term "scheme" for competing algorithms, while referring to FCT as a "method" or "technique," but that invidious distinction, not surprisingly, never caught on. Neither did the term "Flux-Uncorrected Transport," which he used once in a talk.)

My first contribution to helping transform FCT from an algorithm into a method was to derive a formula for the diffusive transport step of SHASTA, i.e., the result of assigning mass to the trapezoids, transporting them, and reassigning it to the mesh. (The original code simply followed the geometric construction.) This formula can be expressed algebraically as

$$\rho_j^{n+1} = \frac{1}{2}Q_-^2\left(\rho_j^n - \rho_{j-1}^n\right) + \frac{1}{2}Q_+^2\left(\rho_{j+1}^n - \rho_j^n\right) + (Q_- + Q_+)\rho_j^n,$$

where

$$Q_\sigma = \frac{1/2 - \sigma u_j \Delta t/\Delta x}{1 + \sigma(u_{j+\sigma} - u_j)\Delta t/\Delta x}, \quad \sigma = \pm 1.$$

For a uniform velocity field, $u_j = u$, it reduces to

$$\rho_j^{n+1} = \rho_j^n - \frac{\varepsilon}{2}\left(\rho_{j+1}^n - \rho_{j-1}^n\right) + \left(\frac{1}{8} + \varepsilon^2\right)\left(\rho_{j+1}^n - 2\rho_j^n + \rho_{j-1}^n\right),$$

which is just Lax–Wendroff plus a zeroth-order (in ε) diffusion with coefficient $1/8$. So, Jay thought, why not use Lax–Wendroff as the transport algorithm even when the flow field is nonuniform, adding diffusion to it and then applying antidiffusion? This worked just as well as SHASTA.

We tried flux-correcting leapfrog, and that worked fine too. Initially we used a diffusion/antidiffusion coefficient of 0.125 because that was what SHASTA used. It turned out that the limit on the Courant number in order to ensure positivity, $\varepsilon < 0.5$, is the same as for SHASTA, although the geometric interpretation no longer holds. Why is $1/8$ the best choice? Wouldn't it be better to use less diffusion sometimes, especially when the flow velocity is very small and less is needed to ensure positivity? I tried running our standard test using different values of the diffusion/antidiffusion coefficient (Fig. 10). It is evident that the best results come from using $1/8$, and that using too much can be as bad as using too little.

We noticed that combining a velocity-dependent antidiffusion with donor cell created a second-order-accurate algorithm formally identical to Lax–Wendroff. If the antidiffusive fluxes are limited with the same prescription that was used in SHASTA the result is a flux-corrected version of donor cell. In this algorithm the diffusion and antidiffusion coefficients vanish when the flow velocity does. In fact, for any value of u flux-corrected donor cell embodies the smallest diffusion coefficient consistent with positivity. But when we tested the new algorithm on square waves we found that the results were inferior to those obtained with flux-corrected Lax–Wendroff or leapfrog. Evidently minimizing the diffusion does not produce the best algorithm.

Jay found the explanation: phase accuracy is more important than amplitude accuracy. This is because the cumulative residual diffusion due to flux limiting results from mashing down the short-wavelength harmonics, which always propagate too

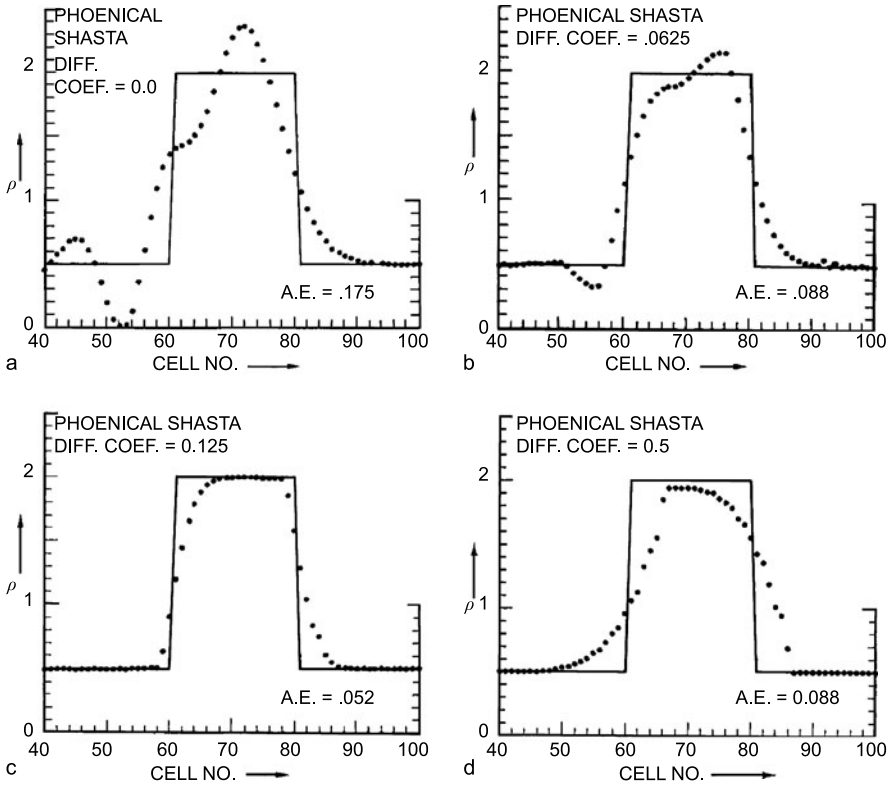

Fig. 10 Result of varying the diffusion/antidiffusion coefficient

slowly or too fast in a finite-difference algorithm. Minimizing the relative phase error works better (because fewer harmonics need mashing) than making the amplification factor as close as possible to unity. In fact, the amplification factor must go to zero for very short wavelengths, or else they wouldn't get mashed. For good results an FCT algorithm should have phase error that is at least second-order (i.e., proportional to k^2 when expanded in powers of wavenumber).

Another lesson we learned from these experiments was that the diffusion and antidiffusion coefficients can be velocity-dependent, provided that the diffusive and antidiffusive fluxes cancel out. This gave us an extra degree of freedom, an additional knob to turn in fine-tuning algorithms.

While musing on an algorithm called hopscotch I had read about in JCP I dreamed up "reversible FCT," which is basically flux-corrected Crank–Nicolson. It applies half the transport step to the old values and half to the new, together with a diffusion/antidiffusion that is also symmetric between the old and new values:

$$\rho_j^T + \frac{\varepsilon}{4}\left(\rho_{j+1}^T - \rho_{j-1}^T\right) + \nu\left(\rho_{j+1}^T - 2\rho_j^T + \rho_{j-1}^T\right)$$
$$= \rho_j^0 - \frac{\varepsilon}{4}\left(\rho_{j+1}^0 - \rho_{j-1}^0\right) + \nu\left(\rho_{j+1}^0 - 2\rho_j^0 + \rho_{j-1}^0\right).$$

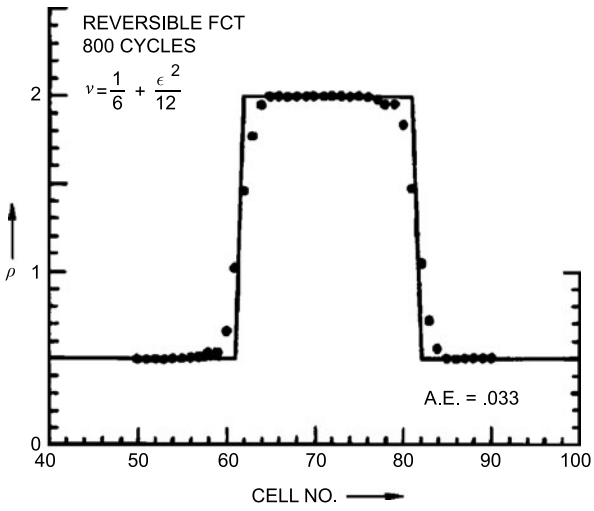

Fig. 11 Reversible FCT

The transported diffused density ρ_j^{TD} is found by adding $\nu(\rho_{j+1}^T - 2\rho_j^T + \rho_{j-1}^T)$ to ρ_j^T. Then the raw antidiffusive flux $\phi_{j+1/2} = \nu(\rho_{j+1}^T - \rho_j^T)$ is corrected with respect to ρ_j^{TD} and reapplied. This algorithm is second-order for any choice of ν because of symmetry. Setting $\nu = 1/6 + \varepsilon^2/12$ makes the phase error fourth-order. With this choice the square-wave test yielded an A.E. of 0.033, the best we had yet found (Fig. 11).

Even without flux correction the underlying transport routine in an FCT algorithm gives better results than conventional algorithms. In a sense it *should*, because conventional algorithms are based on three-point stencils (they involve only the mesh point in question and its two nearest neighbors), while the extra antidiffusion step in FCT introduces information from next-nearest neighbors as well. Can FCT algorithms be made even more accurate by using more complicated stencils?

Thinking about this led me to invent Fourier-transform FCT. Start by Fourier-transforming the density ρ:

$$\rho_j = \sum_{\kappa=1}^{N} \tilde{\rho}_\kappa \exp(2\pi i j \kappa / N).$$

(This is of course implies an N-point stencil, but with fast transforms the computational overhead is acceptable.) Advance each component according to the exact solution of the transformed advection equation:

$$\tilde{\rho}_\kappa(t + \Delta t) = \tilde{\rho}_\kappa(t) \exp(-2\pi i \kappa u \Delta t / \Delta x).$$

Now transform back to x space. The resulting solution has no dissipation and no phase error. On the face of it this algorithm should be error-free, at least for passive advection with a uniform velocity. Indeed, if $\varepsilon = u\Delta t/\Delta x$ is an integer the solution reproduces the analytic solution exactly. There is no need for additional diffusion, antidiffusion, or flux correction.

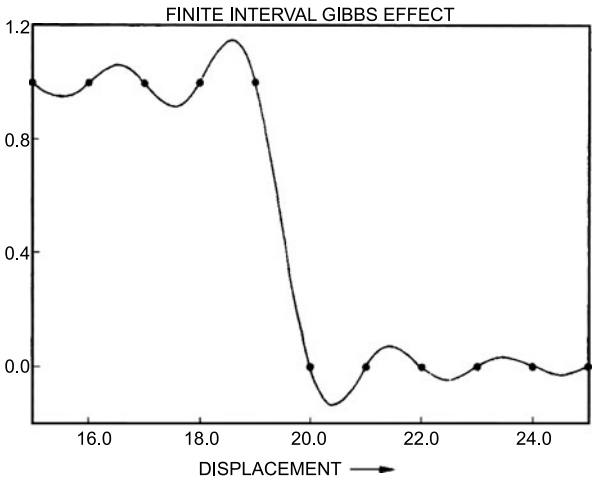

Fig. 12 Continuous function obtained from the discrete Fourier transform of a jump

If $u\Delta t/\Delta x$ is not an integer, however, the solution that results from inverting the Fourier transform has ripples. Jay called this the finite-interval Gibbs effect. These ripples are related to the "window problem," i.e., the impossibility of knowing what goes on between the mesh points. Plotting the inverse of the discrete Fourier transform over the entire interval on which the original discrete function is defined yields a continuous curve. This can be shown to be the smoothest function that passes through the values of the function on the mesh. If the original function contains a sharp discontinuity, this plot not only exhibits the usual Gibbs over- and undershoots at the top and bottom of the jump, but also has wiggles between the mesh points (Fig. 12).

Translating the profile over a fraction of a mesh space exposes these wiggles to view. The Fourier transform thinks a function should behave this way in order to be as smooth as possible, whereas the physics favors one that has as few wiggles as possible. But we know how to fix that: flux-correct it. In other words, add some diffusion, then apply an equal amount of antidiffusion with a flux limiter. Nothing dictates the choice of the coefficient, so we used $\mu = \nu = 1/8$. (Why not? It worked in other algorithms.) The resulting value of 0.022 for the A.E. is the smallest one we ever found (Fig. 13). Thus, at the cost of a higher operation count, Fourier-transform FCT emerged as the optimum FCT algorithm, at least for this test problem.

The three JCP papers focused almost entirely on how FCT worked and on the design of FCT algorithms. In 1976 we contributed an article to the series *Methods of Computational Physics*. It appeared as a chapter in volume 16, which surveyed numerical techniques for plasma physics problems. This article (which perhaps deserves to be designated FCT-4) reviewed our previously published work, but also discussed some of the codes incorporating FCT and the problems to which we had applied them.

When it came to real applications we found that FCT is not perfect. One source of error is something than can be called residual diffusion, which results when antidiffusion fails to completely cancel diffusion.

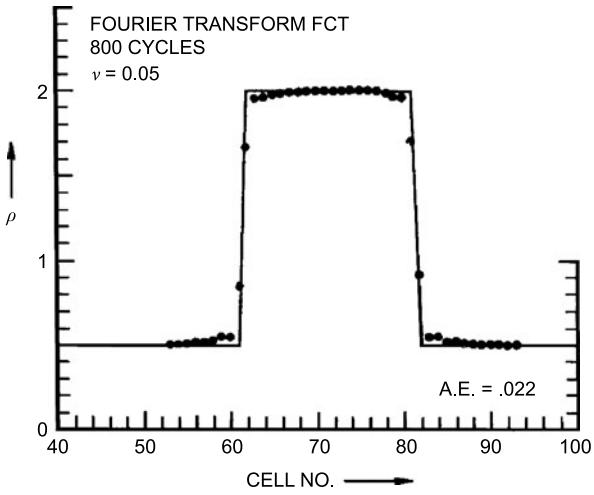

Fig. 13 The optimum FCT algorithm

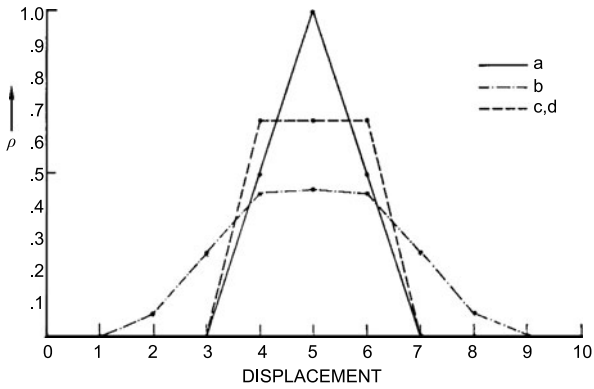

Fig. 14 Result of 20 repetitions to profile (*a*) of diffusion and (*b*) explicit; (*c*) phoenical; (*d*) implicit antidiffusion with strong flux limiting, using coefficients $\mu = \nu = 0.2$

The most dramatic manifestation of this is "clipping." An extremum loses a little bit of amplitude each timestep, even if it isn't being advected across a grid, because diffusion squashes the extremum down and strong flux limiting doesn't allow the antidiffusion to push it back up. Ultimately the peak changes into a characteristic flat-topped structure, which we called a plateau. For example, an initially sharp maximum subjected to repeated diffusion and antidiffusion operations gradually flattens out until it forms a plateau three points across (Fig. 14), which is stable.

Figure 14 shows that this flattening is less severe with phoenical or implicit antidiffusion than with the original explicit form of antidiffusion, in which the raw flux is calculated from the diffused profile. The reason is obvious. If T stands for the transport operation, D stands for a three-point centered second difference with co-

efficient ν, and A represents the same operation with coefficient $\mu = -\nu$, the three versions of FCT can be represented symbolically as

$$\begin{aligned}
&\text{Explicit:} \quad \rho^1 = (1 + A)\rho^{TD} = (1 + A)(1 + T + D)\rho^0;\\
&\text{Phoenical:} \quad \rho^1 = \big[(1 + A)(1 + T) + D\big]\rho^0;\\
&\text{Implicit:} \quad \rho^1 = (1 + D)^{-1}\rho^{TD} = (1 + D)^{-1}(1 + T + D)\rho^0.
\end{aligned}$$

It is clear that in the limit $T \to 0$ phoenical and implicit FCT reduce to the identity operation, or would except for the action of the flux limiter, but explicit FCT does not. (This is of course why phoenical antidiffusion was invented.)

In an effort to eliminate clipping we tried a number of different flux limiters. One of my early attempts was the one-sided flux limiter. This involved changing the flux limiter so that maxima could grow on positive profiles but minima could not (and vice-versa for negative profiles). The resulting algorithm preserved positivity, but gave rise—unsurprisingly—to one-sided ripples.

Another idea of mine that didn't work out was called "flux-limited diffusion." The innovation here was to apply diffusion only where needed to prevent extrema from growing relative to their original values, rather than put it in everywhere and remove it where it was not needed. The test results, however, were disappointing. Square waves propagated using flux-limited diffusion were badly eroded. The trouble with this approach was that it failed to ensure high phase accuracy in the underlying transport scheme. Ultimately we decided to stick with strong flux limiting. It was easier to live with the symptoms of the disease than with the side effects of the cures.

Another form of residual diffusion is more subtle. If the antidiffusion coefficient is smaller than the diffusion coefficient, some residual diffusion is present even in the absence of flux limiting. The algorithm with fourth-order phase accuracy described in FCT-3 has an amplification factor that just barely—by less than 0.5%—exceeds unity. (We didn't know this until Phil Colella pointed it out years later.) Because of this it was found that codes yielded the best results when the antidiffusion coefficient was reduced slightly by multiplying by a factor called a "mask." (The name arose because the correction was applied using MASK, a Fortran IV bit-manipulating instruction.)

An annoying but fairly innocuous departure from realism arises because the flux limiter stops dispersive ripples from growing on the slopes of hills or valleys only when they are about to form new extrema. This leaves flat "terraces" on the slopes, which I like to think of as the ghosts of departed ripples. Improving the phase accuracy of the algorithm helps reduce terracing, but the only real cure is to use a more elaborate form of flux limiting that takes into account second derivatives of the profile.

When we started using FCT for two-dimensional problems a new and much more serious question arose: How could we generalize the flux limiter to multidimensions? Should all the fluxes be corrected in one sweep? In applying antidiffusive fluxes in, say, the x direction, should we worry about extrema only along that axis, or should we look at all points in the neighborhood? The coding was tortuous, and every prescription we tried seemed to fail in some combination of circumstances.

Finally we gave up and decided to use coordinate splitting. That is, on each timestep we treated each coordinate independently, carrying out 1-D transport and 1-D flux limiting first in the x direction and then in y. Symbolically this can be written, e.g.,

$$\rho^{TD} = (1 + T_x + D_x)(1 + T_y + D_y)\rho^0;$$
$$\rho^1 = (1 + A_x)(1 + A_y)\rho^{TD}.$$

Obviously this introduces spurious terms, e.g., $A_x A_y$. Most of the time they do no great harm because the errors tend to cancel out, but there are some situations where splitting creates unphysical effects. One example is when plateau formation occurs in both coordinate directions at the same time, for example on a hilltop. The result is that contours of constant density (or pressure, etc.) become square. I got very tired of going to meetings where I had to explain why my code generated square fireballs.

5 The Next Generation

Despite all our efforts we never found a way around the problem of creating a multidimensional flux limiter. Eventually I decided that it was insoluble, and I told every colleague who expressed an interest in it not to waste his time. As is well known, the problem turned out not to be insoluble. Likewise, the "pioneering idea of blending high- and low-order discretizations," cited on the FCT-30 web page, was not part of the original concept. It was a later development, due to the same individual who created the multidimensional flux limiter, Steve Zalesak. But that is his story to tell.

References

1. Book, D.L., Boris, J.P., McDonald, B.E., Wagner, C.E.: SHASTA, a transport algorithm that works. In: Proc. Symposium on High-Altitude Nuclear Effects, Stanford, CA, August 10–12 (1971)
2. Book, D.L., Boris, J.P.: A transport algorithm that works. NRL Reports of Progress (Nov. 1971), p. 1
3. Boris, J.P., Book, D.L.: Flux-corrected transport: I. SHASTA, a fluid transport algorithm that works. J. Comput. Phys. **11**, 38–69 (1973) [FCT-1]
4. Book, D.L., Boris, J.P., Hain, K.: Flux-corrected transport: II. Generalizations of the method. J. Comput. Phys. **18**, 248–283 (1975) [FCT-2]
5. Boris, J.P., Book, D.L.: Flux-corrected transport: III. Minimal-error FCT algorithms. J. Comput. Phys. **20**, 397–431 (1976) [FCT-3]
6. Boris, J.P., Book, D.L.: Solution of continuity equations by the method of flux-corrected transport. In: Alder, B., Fernbach, S., Rotenberg, M., Killeen, J. (eds.) Methods in Computational Physics, vol. 16. Academic Press, New York (1976)