NAVAL RESEARCH LABORATORY
Washington, DC 20375-5320
NRL/MR/6410-93-7192

LCPFCT – Flux-Corrected Transport Algorithm for Solving Generalized Continuity Equations

JAY P. BORIS
ALEXANDRA M. LANDSBERG
ELAINE S. ORAN
JOHN H. GARDNER

Laboratory for Computational Physics and Fluid Dynamics

April 16, 1993

LCPFCT – A Flux-Corrected Transport Algorithm for Solving Generalized Continuity Equations

Jay P. Boris, Alexandra M. Landsberg, Elaine S. Oran and John H. Gardner

Laboratory for Computational Physics and Fluid Dynamics

U.S. Naval Research Laboratory, Washington DC

ABSTRACT

Flux-Corrected Transport has proven to be an accurate and easy to use algorithm to solve nonlinear, time-dependent continuity equations of the type which occur in fluid dynamics, reactive, multiphase, and elastic plastic flows, plasma dynamics, and magnetohydrodynamics. This report updates and supersedes a previous report entitled "Flux-Corrected Transport Modules for Solving Generalized Continuity Equations." It can be used as a user manual for the subroutines and test programs included in the appendices. The entire LCPFCT library in its most recent form is presented and discussed in detail. There are, in addition, discussions of more general topics such as the application of physical boundary conditions, physical positivity and numerical diffusion which help to put the numerical aspects of this subroutine library in context.

TABLE OF CONTENTS

1. INTRODUCTION

This report explains and documents a group of subroutines for solving generalized continuity equations of the form

$$\frac{\partial \rho}{\partial t} = -\frac{1}{r^{\alpha-1}}\frac{\partial}{\partial r}(r^{\alpha-1}\rho v) - \frac{1}{r^{\alpha-1}}\frac{\partial}{\partial r}(r^{\alpha-1}D_1) + C_2\frac{\partial D_2}{\partial r} + D_3 \ . \tag{1.1}$$

These subroutines, collectively referred to by the name of the main program, LCPFCT, use one of the latest one-dimensional Flux-Corrected Transport (FCT) algorithms with fourth-order phase accuracy and minimum residual diffusion. The program loops vectorize to take full advantage of vector architectures and run equally well on scalar and superscalar computers. The use of internal temporary memory is quite minimal, limited to about thiry short one-dimensional arrays, and is arranged to maximize readability and efficient program execution. A rather general capability to handle the source terms in Eq. (1) has been provided so that coupled sets of multidimensional nonlinear continuity equations, such as those for ideal compressible fluid dynamics and reactive flows, can be solved using the routines presented here.

LCPFCT itself can treat one-dimensional, Cartesian, cylindrical, or spherical , and generalized nozzle coordinates. A flexible set of boundary conditions for each equation can be selected by the appropriate choice of the arguments to the subroutine calls. In addition to inflow, outflow, and reflecting wall conditions in several coordinate systems, there is an option for periodic boundary conditions. Using this version of LCPFCT, multidimensional problems may be solved by timestep-splitting techniques. The computational grid can be nonuniform and, in addition, can move during the course of a timestep, enabling us to do Lagrangian and sliding rezone calculations. The programs produce a positive, conservative interpolation when the fluid velocity is zero but the grid moves, which is an important test of the gridding.

The important properties of FCT are that it is a high-order, monotone, conservative, positivity preserving algorithm. This means that the algorithm is accurate and resolves steep gradients, allowing grid scale numerical resolution. When a convected quantity such as a density is initially positive, it remains positive and no new maxima or minima are introduced due to numerical errors in the convection process. These are properties that are extremely important for most problems of practical interest. Table 1 presents an overview of FCT algorithm developments. More background, description, and historical material may be found in Boris (1971), Boris and Book (1976), Book and Boris (1981), and Oran and Boris (1987).

The material presented here is an update and expansion of the ETBFCT programs described by Boris (1976). There are several fundamental differences between LCPFCT

Table 1.1 History of Development of FCT Algorithms

1971	Basic nonlinear, monotone algorithm (Boris)
1976	Adaptation of FCT to general finite-difference algorithms (Boris and Book)
1976	Optimization for vector and parallel processing (Boris)
1979	Fully multidimensional FCT and generalization to use with arbitrary high- and low-order algorithms (Zalesak)
1985	Finite-Element FCT on triangle-based grids (Löhner)
1986	Implicit FCT (Patnaik)
1991	Arbitrary nonorthogonal FCT (Fyfe and Patnaik)
1992	General curved boundary FCT (Landsberg and Boris)

and ETBFCT. First, in LCPFCT, variables are defined on cell centers instead of cell vertices, a relatively small change. Second, ETBFCT was written as a single subprogram, with a number of entries whereas LCPFCT is a series of independent subroutines which communicate through named common blocks. Finally, additional subroutines have been added to increase the flexibility and ease of using LCPFCT.

LCPFCT is written in Fortran. Complete program listings and four test programs are given in the appendices. Appendix A contains a complete listing of the series of subroutines which in their entirety constitute LCPFCT. Appendix B contains a constant velocity convection test problem, LCPFCT Test #1, with the driver program and sample results in tabulated form that can be used to check the code. Appendix C contains a progressing shock test problem, LCPFCT Test #2, and selected outputs for comparison. Appendix C also has an interface program called GASDYN which combines the calls to source generating routines, velocity and boundary condition routines, and the basic continuity equation module LCPFCT. GASDYN couples the set of nonlinear continuity equations to solve gasdynamics one row at a time and is used in LCPFCT Tests #2, #3, and #4. Appendix D contains the program and selected outputs for the one-dimensional bursting diaphragm problem, LCPFCT Test #3. This example illustrates the variable grid features of the LCPFCT routines by switching into an expanding system of grid coordinates to capture the expected similarity solution. Appendix E contains a two-dimensional "muzzle flash" test problem, LCPFCT Test #4, with sample output that can be used to verify the users version of the code. This fourth test illustrates the use of simple outflow boundary conditions and shows how to construct programs with relatively complex geometries.

2. NUMERICAL BACKGROUND

2.1 Positivity and Accuracy

Good resolution of steep gradients is important in many problems we need to solve. It is important in reactive flows, where the gradients at detonation fronts, flame fronts, and at interfaces in multiphase flows must be accurately represented. Flame speeds depend on steep species gradients, as do the local energy release profiles. It is important in simulations of shocks, particularly when they collide or interact with other steep gradients. High resolution of shear flows is also very important since vortex stretching and shear steepening both produce steep local gradients in the flow.

Positivity is a property satisfied by the continuity equation. When the density $\rho(r,t)$ in Eq. (1.1) is everywhere positive and the source terms are zero, it is a mathematical consequence of the continuity equation and an obvious physical property of the flow that the density can never become negative anywhere – regardless of the velocity field specified. To retain this mathematical and physical property in numerical convection through an Eulerian grid involves a certain amount of numerical diffusion. This numerical diffusion arises as a consequence of the physical requirements that the profiles being convected remain stable while remaining positive. Numerical diffusion is an inherent problem in Eulerian convection, and unless controlled, it can invalidate numerical calculations using linear algorithms unless they have very fine computational meshes.

Figure 2.1 shows how numerical diffusion enters the first-order *upwind* algorithm (see, for example, Oran and Boris, 1987). Consider a discontinuity, at $x = 0$ at time $t = 0$, that moves at a constant velocity from left to right. The velocity, v, the timestep, Δt, and the computational cell size, Δx, are chosen such that $v\Delta t/\Delta x = 1/3$ in the figure. This means that the actual physical discontinuity travels one third of a cell per timestep. The solution obtained using the linear upwind algorithm (sometimes called *donor-cell*) is given by the solid line. The "upwind" finite-difference formula is a simple linear interpolation

$$\rho_i^{n+1} \;=\; \rho_i^n - \frac{v\Delta t}{\Delta x}(\rho_i^n - \rho_{i-1}^n)\;. \tag{2.1}$$

If the $\{\rho_i\}$ are positive at some time $t = n\ \Delta t$ and

$$\left|\frac{v\Delta t}{\Delta x}\right| \;\leq\; 1 \tag{2.2}$$

in each cell, the new density values $\{\rho_i^{n+1}\}$ at time $t = (n+1)\Delta t$ are also positive. The price for guaranteed positivity in this linear algorithm is a severe nonphysical spreading of the discontinuity which should be located at $x = vt$.

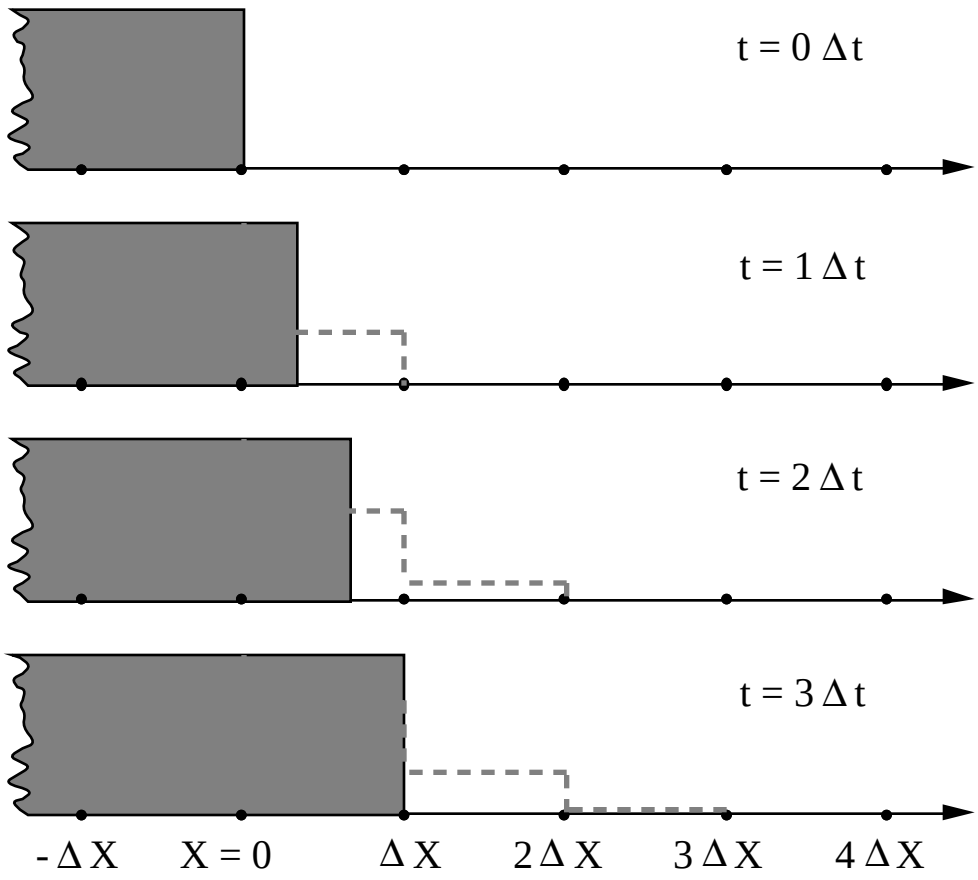

Figure 2.1 The results of convecting a discontinuity with the highly diffusive, first order upwind algorithm. The velocity is 1/3 of a cell per timestep. The solid lines are the exact profile, which coincides with the numerical solution at $t = 0$. The heavy dashed line is the numerical solution. Note the diffusive precursor moves at once cell per timestep regardless of the speed of the flow.

In the example shown in Figure 2.1, the initial discontinuity erodes rapidly. This process looks like physical diffusion, but it arises here from numerical errors. The numerical diffusion occurs because material that has just entered a cell, and should still be near the left boundary, is smeared over the whole cell when the transported fluid elements are interpolated linearly back onto the Eulerian grid. Higher-order approximations to the convective derivatives are required to reduce this diffusion.

Now consider a three-point explicit finite-difference formula for advancing $\{\rho_i^n\}$ one timestep to $\{\rho_i^{n+1}\}$,

$$\rho_i^{n+1} \;=\; a_i \rho_{i-1}^n + b_i \rho_i^n + c_i \rho_{i+1}^n \;. \tag{2.3}$$

This general form includes the first-order upwind algorithm and other common algorithms.

If Δx and Δt are constants, Eq. (2.3) can be rewritten in a form that guarantees conservation,

$$\begin{aligned}\rho_i^{n+1} &= \rho_i^n - \frac{1}{2}\left[\epsilon_{i+\frac{1}{2}}(\rho_{i+1}^n + \rho_i^n) - \epsilon_{i-\frac{1}{2}}(\rho_i^n + \rho_{i-1}^n)\right] \\ &\quad + \left[\nu_{i+\frac{1}{2}}(\rho_{i+1}^n - \rho_i^n) - \nu_{i-\frac{1}{2}}(\rho_i^n - \rho_{i-1}^n)\right] ,\end{aligned} \tag{2.4}$$

where

$$\epsilon_{i+\frac{1}{2}} \equiv v_{i+\frac{1}{2}} \frac{\Delta t}{\Delta x} . \tag{2.5}$$

The $\{\nu_{i+\frac{1}{2}}\}$ are nondimensional numerical diffusion coefficients which appear as a consequence of considering adjacent grid points. Conservation of ρ in Eq. (2.4) also constrains the coefficients a_i, b_i, and c_i in Eq. (2.3) by the condition

$$a_{i+1} + b_i + c_{i-1} = 1 . \tag{2.6}$$

Positivity of $\{\rho_i^{n+1}\}$ for all possible positive profiles $\{\rho_i^n\}$ requires that $\{a_i\}$, $\{b_i\}$, and $\{c_i\}$ be positive for all i.

Matching corresponding terms in Eqs. (2.4) and (2.3) gives

$$\begin{aligned} a_i &\equiv \nu_{i-\frac{1}{2}} + \frac{1}{2}\epsilon_{i-\frac{1}{2}} , \\ b_i &\equiv 1 - \frac{1}{2}\epsilon_{i+\frac{1}{2}} + \frac{1}{2}\epsilon_{i-\frac{1}{2}} - \nu_{i+\frac{1}{2}} - \nu_{i-\frac{1}{2}} , \\ c_i &\equiv \nu_{i+\frac{1}{2}} - \frac{1}{2}\epsilon_{i+\frac{1}{2}} . \end{aligned} \tag{2.7}$$

If the $\{\nu_{i+\frac{1}{2}}\}$ are positive and large enough, they ensure that the $\{\rho_i^{n+1}\}$ are positive. The positivity conditions derived from Eqs. (2.7) are

$$\begin{aligned} |\epsilon_{i+\frac{1}{2}}| &\le \frac{1}{2} , \\ \frac{1}{2} \ge \nu_{i+\frac{1}{2}} &\ge \frac{1}{2}\,|\epsilon_{i+\frac{1}{2}}| , \end{aligned} \tag{2.8}$$

for all i. Thus the condition in Eq. (2.8) for *positivity* leads directly to numerical diffusion in addition to the desired convection,

$$\begin{aligned}\rho_i^{n+1} &= \rho_i^n + \nu_{i+\frac{1}{2}}(\rho_{i+1}^n - \rho_i^n) - \nu_{i-\frac{1}{2}}(\rho_i^n - \rho_{i-1}^n) \\ &\quad + \text{convection} ,\end{aligned} \tag{2.9}$$

where Eq. (2.8) holds. This first-order numerical diffusion rapidly smears a sharp discontinuity. Godunov has shown rather generally that linear second-order algorithms cannot

uphold physical positivity. If algorithms are used with $\nu_{i+\frac{1}{2}} < \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$, positivity is not necessarily destroyed but can no longer be guaranteed. In practice, the positivity conditions are almost always violated by strong shocks and discontinuities unless the inequalities stated in Eq. (2.8) hold. Nevertheless, the numerical diffusion implied by Eq. (2.8) is unacceptable. The diffusion coefficient $\{\nu_{i+\frac{1}{2}}\}$ cannot be zero, however, because the explicit three-point formula, Eq. (2.4), is subject to a numerical stability problem if it is zero. Finite-difference methods which are higher than first order, such as the Lax-Wendroff (1964) methods, reduce the numerical diffusion but sacrifice assured positivity. This apparent dilemma can only be resolved by using a nonlinear method to integrate the continuity equations.

To examine the problem of stability and positivity, we consider a stability analysis. Consider convecting test functions of the form

$$\rho_i^n \equiv \rho_o^n e^{\mathrm{i}i\beta} , \tag{2.10}$$

where

$$\beta \equiv k\,\Delta x = \frac{2\pi\,\Delta x}{\lambda} , \tag{2.11}$$

and i indicates $\sqrt{-1}$. Substituting this solution into Eq. (2.4) gives

$$\rho_o^{n+1} = \rho_o^n\left[1 - 2\nu(1-\cos\beta) - \mathrm{i}\epsilon\sin\beta\right] , \tag{2.12}$$

where we assume that

$$\begin{aligned} \{\nu_{i+\frac{1}{2}}\} &= \nu \\ \{\epsilon_{i+\frac{1}{2}}\} &= \epsilon . \end{aligned} \tag{2.13}$$

The exact theoretical solution to this linear problem is

$$\rho_o^{n+1}|_{\text{exact}} = \rho_o^n e^{-\mathrm{i}kv\Delta t} . \tag{2.14}$$

Therefore the difference between the exact solution and Eq. (2.12) is the numerical error generated at each timestep.

The amplification factor was defined as

$$A \equiv \frac{\rho_o^{n+1}}{\rho_o^n} , \tag{2.15}$$

and an algorithm is always linearly stable if

$$|A|^2 \le 1 . \tag{2.16}$$

From Eq. (2.12),

$$|A|^2 \;=\; 1-(4\nu-2\epsilon^2)(1-\cos\beta)+(4\nu^2-\epsilon^2)(1-\cos\beta)^2\;, \tag{2.17}$$

which ought to be less than unity for all permissible values of β between 0 and π. In general, $\nu > \frac{1}{2}\epsilon^2$ ensures stability of the linear convection algorithm for any Fourier harmonic of the disturbance, provided that Δt is chosen so that $|\epsilon| \leq 1$. This stability condition is a factor of two less stringent than the positivity conditions $|\epsilon| \leq \frac{1}{2}$. When $\nu > \frac{1}{2}$, there are combinations of ϵ and β where $|A|^2 > 1$, for example $\epsilon = 0$ with $\beta = \pi$. Thus the range of acceptable diffusion coefficients is quite closely prescribed,

$$\frac{1}{2} \;\geq\; \nu \;\geq\; \frac{1}{2}|\epsilon| \;\geq\; \frac{1}{2}\epsilon^2\;. \tag{2.18}$$

Even the minimal numerial diffusion required for linear stability, $\nu = \frac{1}{2}\epsilon^2$, may be substantial when compared to the physically correct diffusion effects such as thermal conduction, molecular diffusion, or viscosity. Figure 2.2 shows the first few timesteps from the same test problem as in Figure 2.1, but using $\nu = \frac{1}{2}\epsilon^2$ rather than $\nu = \frac{1}{2}\epsilon$ required for positivity. The profile spreads only one third as much as in the previous case where positivity was assured linearly, but a numerical precursor still reaches two cells beyond the correct discontinuity location. Furthermore, the overshoot between $x = -\Delta x$ and $x = 0$ in Figure 2.2 is a consequence of underdamping the solution. The loss of *monotonicity* indicated by the overshoot can be as bad as violating positivity. A new, nonphysical maximum in ρ has been introduced into the solution. When the convection algorithm is stable but not positive, the numerical diffusion is not large enough to mask either numerical dispersion or the Gibbs phenomenon arising near sharp gradients so the solution is no longer necessarily monotone. New ripples, that is, new maxima or minima, are introduced numerically.

2.2 Principles of Flux-Corrected Transport

From the discussion above and the work of Godunov (1959), the requirements of positivity and accuracy seem to be mutually exclusive. *Nonlinear monotone methods* were invented to circumvent this dilemma. These methods use the stabilizing $\nu = \frac{1}{2}\epsilon^2$ diffusion where monotonicity is not threatened, and increase ν to values approaching $\nu = \frac{1}{2}|\epsilon|$ when required to assure that the solution remains monotone. Different criteria are imposed in the same timestep at different locations on the computational grid according to the local profiles of the physical solution. *The dependence of the local smoothing coefficients ν on the solution profile makes the overall algorithm nonlinear.*

To prevent negative values of ρ which could arise from dispersion or Gibbs errors, a minimum amount of numerical diffusion must be added to assure positivity and stability

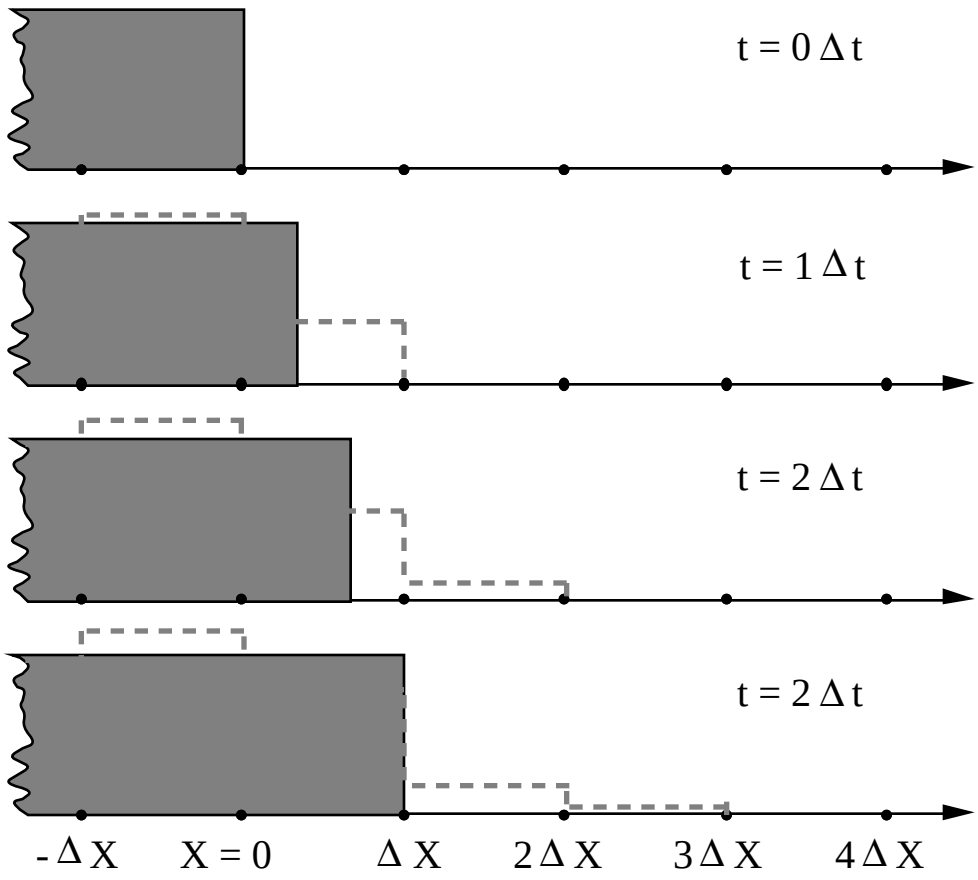

Figure 2.2 Results of convecting a discontinuity using an algorithm with enough diffusion to maintain stability, but not enough to hide the effects of dispersion. Note the growing nonphysical overshoot behind the actual discontinuity and the diffusive numerical precursor at times after $t = 0$ in the numerical solution (heavy dashed line).

at each timestep. We write this minimal diffusion as

$$\nu \approx \frac{|\epsilon|}{2}\,(c + |\epsilon|) \tag{2.19}$$

where c is a *clipping factor*, $0 \le c \le 1 - |\epsilon|$, that controls how much extra diffusion must be added to ensure positivity over that required for stability, $\epsilon^2/2$. In the vicinity of steep discontinuities, $c \approx 1 - |\epsilon|$, and in smooth regions away from local maxima and minima, $c \approx 0$.

Over the last 20 years, monotone algorithms have been shown to be a reliable, robust way to calculate convection. The first specifically monotone, positivity-preserving technique was the Flux-Corrected Transport (FCT) algorithm developed at NRL, as discussed,

for example, in Boris (1971) and Boris and Book (1973, 1976). Other early monotone methods employing nonlinear flux limiters were proposed by van Leer (1973, 1979), and Harten (1974, 1983). There has been extensive work on monotone methods during the last ten years, some of which is described in the following references, Colella and Woodward (1984) and Woodward and Colella (1984), Baer (1986), and Rood (1987). A characteristic of these methods generally distinguishing them from FCT is their use of a Riemann solver to determine the fluxes of mass momentum and energy for gas dynamics. Their use of nonlinear limiting formulae on these fluxes to calculate the clipping factor c above, is very much like FCT. Research on monotone methods related to the FCT approach without a Reiman solver has also continued to the present, for example by Odstrcil (1990), Leonard and Niknafs (1990), Nessyahu and Tadmor (1990), and Lafon and Osher (1992). Zalesak (1979, 1981), Löhner (1987), Patnaik, et al. (1987), DeVore (1989, 1991), Fyfe and Patnaik (1991), and Landsberg and Boris (1992) have developed various generalizations and modifications of FCT designed to improve its performance in multidimensions and to represent complex geometry.

We now rewrite the explicit three-point approximation to the continuity equation given in Eq. (2.3) to determine provisional values, $\{\tilde{\rho}_i\}$, from the previous timestep or "old" values, $\{\rho^o{}_i\}$,

$$\tilde{\rho}_i \;=\; a_i\rho^o_{i-1} + b_i\rho^o_i + c_i\rho^o_{i+1} \;. \tag{2.20}$$

Again, Eq. (2.6) must be satisfied for conservation and $\{a_i\}$, $\{b_i\}$, and $\{c_i\}$ must all be greater than or equal to zero to assure positivity.

Equation (2.20), in conservative form, again becomes

$$\begin{aligned}\tilde{\rho}_i \;&=\; \rho^o_i - \frac{1}{2}\left[\epsilon_{i+\frac{1}{2}}(\rho^o_{i+1}+\rho^o_i) - \epsilon_{i-\frac{1}{2}}(\rho^o_i+\rho^o_{i-1})\right] \\ &\quad + \left[\nu_{i+\frac{1}{2}}(\rho^o_{i+1}-\rho^o_i) - \nu_{i-\frac{1}{2}}(\rho^o_i-\rho^o_{i-1})\right] \\ &=\; \rho^o_i - \frac{1}{\Delta x}\left[f_{i-\frac{1}{2}} - f_{i+\frac{1}{2}}\right] \;.\end{aligned} \tag{2.21}$$

The values of variables at interface $i+\frac{1}{2}$ are averages (possibly unequally weighted) of values at cells $i+1$ and i, and the values at $i-\frac{1}{2}$ are averages of values at cells i and $i-1$. At every cell i, the $\tilde{\rho}_i$ differs from ρ^o_i as a result of the inflow and outflow fluxes of ρ, denoted by $\{f_{i\pm\frac{1}{2}}\}$ across the cell boundaries. The fluxes are successively added and subtracted along the array of densities $\{\rho^o_i\}$ so that the overall conservation of ρ is satisfied by construction. Summing all the provisional densities gives the sum of the old densities. The expressions involving $\epsilon_{i\pm\frac{1}{2}}$ are called the *convective fluxes*.

By comparing Eq. (2.21) and (2.20), we obtain the conditions relating the a, b, and c's to the ϵ's and ν's, essentially as in Eq. (2.7). In Eq. (2.21), the $\{\nu_{i+\frac{1}{2}}\}$ are dimensionless

diffusion coefficients included to ensure positivity of the provisional values $\{\tilde{\rho}_i\}$. The positivity condition for the provisional $\{\tilde{\rho}_i\}$ is given in Eq. (2.8).

However, after Eq. (2.20) is imposed, two of the three coefficients in Eq. (2.21) are still to be determined. One of these sets of coefficients must ensure an accurate representation of the mass flux terms. Thus

$$\epsilon_{i+\frac{1}{2}} \;=\; v_{i+\frac{1}{2}}\;\frac{\Delta t}{\Delta x}\;, \tag{2.22}$$

where, $\{v_{i+\frac{1}{2}}\}$ is the fluid velocity approximated at the cell interfaces. The other set of coefficients, $\{\nu_{i+\frac{1}{2}}\}$, are chosen to maintain positivity and stability.

The provisional values $\tilde{\rho}_i$ must be strongly diffused to ensure positivity. If $\nu_{i+\frac{1}{2}} = \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$ in Eq. (2.8), we have the diffusive, first-order upwind algorithm. A correction in FCT to remove this strong diffusion involves an additional *antidiffusion* stage,

$$\rho_i^n \;=\; \tilde{\rho}_i - \mu_{i+\frac{1}{2}}(\tilde{\rho}_{i+1} - \tilde{\rho}_i) + \mu_{i-\frac{1}{2}}(\tilde{\rho}_i - \tilde{\rho}_{i-1})\;, \tag{2.23}$$

in the algorithm to get the new values of $\{\rho_i^n\}$. Here $\{\mu_{i+\frac{1}{2}}\}$ are positive *antidiffusion coefficients*. Antidiffusion reduces the strong diffusion implied by Eq. (2.8), but also reintroduces the possibility of negative values or nonphysical overshoots in the "corrected" profile. If the values of $\{\mu_{i+\frac{1}{2}}\}$ are too large, the new solution $\{\rho_i^n\}$ will be unstable numerically.

To obtain a positivity-preserving algorithm, we modify the antidiffusive fluxes in Eq. (2.23) by a process that we call *flux correction*. The antidiffusive fluxes,

$$f_{i+\frac{1}{2}}^{ad} \;\equiv\; \mu_{i+\frac{1}{2}}\;(\tilde{\rho}_{i+1} - \tilde{\rho}_i)\;, \tag{2.24}$$

appearing in Eq. (2.23) are corrected (*limited*) as described below to ensure positivity and stability.

The biggest choice of the antidiffusion coefficients $\{\mu_{i+\frac{1}{2}}\}$ that still guarantees positivity linearly is

$$\mu_{i+\frac{1}{2}} \;\approx\; \nu_{i+\frac{1}{2}} - \frac{1}{2}\;|\epsilon_{i+\frac{1}{2}}|\;. \tag{2.25}$$

However, this is not large enough. To reduce the residual diffusion $(\nu - \mu)$ even further, the flux correction must be nonlinear, depending on the actual values of the density profile $\{\tilde{\rho}_i\}$.

The idea behind the nonlinear flux-correction formula is as follows: Suppose the density $\tilde{\rho}_i$ at grid point i reaches zero while its neighbors are positive. Then the second derivative is locally positive and any antidiffusion would force the minimum density value

$\tilde{\rho}_i = 0$ to be negative. Because this cannot be allowed on physical grounds, the antidiffusive fluxes should be limited so minima in the profile are made no deeper by the antidiffusive stage of Eq. (2.23). Because the continuity equation is linear, we could equally well solve for $\{-\rho_i^n\}$. Hence, we also must require that antidiffusion not make the maxima in the profile any larger. These two conditions form the basis for FCT and a central role in other monotone methods. *The antidiffusion stage should not generate new maxima or minima in the solution, nor accentuate already existing extrema.*

This qualitative idea of a nonlinear filtering can be quantified. The new values $\{\rho_i^n\}$ are given by

$$\rho_i^n \;=\; \tilde{\rho}_i - f^c_{i+\frac{1}{2}} + f^c_{i-\frac{1}{2}} \;, \tag{2.26}$$

where the corrected fluxes $\{f^c_{i+\frac{1}{2}}\}$ satisfy

$$f^c_{i+\frac{1}{2}} \;\equiv\; S \cdot \max\Big\{0,\; \min\big[S\cdot(\tilde{\rho}_{i+2} - \tilde{\rho}_{i+1}),\; |f^{ad}_{i+\frac{1}{2}}|,\; S\cdot(\tilde{\rho}_i - \tilde{\rho}_{i-1})\big]\Big\} \;. \tag{2.27}$$

Here $|S| = 1$ and sign $S \equiv$ sign $(\tilde{\rho}_{i+1} - \tilde{\rho}_i)$.

To see what this flux-correction formula does, assume that $(\tilde{\rho}_{i+1} - \tilde{\rho}_i)$ is greater than zero. Then Eq. (2.27) gives either

$$\begin{aligned} f^c_{i+\frac{1}{2}} \;&=\; \min\big[(\tilde{\rho}_{i+2} - \tilde{\rho}_{i+1}),\; \mu_{i+\frac{1}{2}}(\tilde{\rho}_{i+1} - \tilde{\rho}_i),\; (\tilde{\rho}_i - \tilde{\rho}_{i-1})\big] \;\text{ or} \\ f^c_{i+\frac{1}{2}} \;&=\; 0 \;, \end{aligned} \tag{2.28}$$

whichever is larger. The "raw" antidiffusive flux, $f^{ad}_{i+\frac{1}{2}}$ given in Eq. (2.24), always tends to decrease ρ_i^n and to increase ρ_{i+1}^n. The flux-limiting formula ensures that the corrected flux cannot push ρ_i^n below ρ_{i-1}^n, which would produce a new minimum, or push ρ_{i+1}^n above ρ_{i+2}^n, which would produce a new maximum. Equation (2.27) is constructed to take care of all cases of sign and slope.

The formulation of an FCT transport algorithm therefore consists of the following four sequential stages:

1. Compute the transported and diffused values $\tilde{\rho}_i$ from Eq. (2.21), where the $\nu_{i+\frac{1}{2}} > \frac{1}{2}|\epsilon_{i+\frac{1}{2}}|$ to satisfy monotonicity. Add in any additional source terms, for example, $-\nabla P$.

2. Compute the raw antidiffusive fluxes from Eq. (2.24).

3. Correct or limit these fluxes using Eq. (2.27) to assure monotonicity.

4. Perform the indicated antidiffusive correction through Eq. (2.26).

Stages 3 and 4 are the new components introduced by FCT. There are many modifications of this prescription that accentuate various properties of the solution. Some of these are summarized in Boris and Book (1976), by Zalesak (1979, 1981), and more recently in Book, et al. (1991).

3. THE LCPFCT ALGORITHM

We now discuss the program LCPFCT, implemented as a Fortran subroutine for solving the continuity equation. LCPFCT is used in combination with calls to a number of auxiliary subroutines for defining the computational grid, the velocity dependent factors, the various source terms in the equations being solved, and the boundary conditions. This is an updated version of the program ETBFCT (Boris, 1976) and is available on request. The programs are short and complete program listings also appear in the appendices.

LCPFCT implements an explicit solution of the general one-dimensional continuity equation, Eq. 1.1, which is reprinted just below,

$$\frac{\partial \rho}{\partial t} = -\frac{1}{r^{\alpha-1}}\frac{\partial}{\partial r}(r^{\alpha-1}\rho v) - \frac{1}{r^{\alpha-1}}\frac{\partial}{\partial r}(r^{\alpha-1}D_1) + C_2\frac{\partial D_2}{\partial r} + D_3. \tag{1.1}$$

The current implementation includes provisions for a spatially variable and moving grid. Additional source terms are included by means of the terms D_1, D_2, and D_3. Different one-dimensional geometries may be selected through variation of an input integer α where $\alpha = 1$ is Cartesian or planar geometry, $\alpha = 2$ is cylindrical geometry, and $\alpha = 3$ is spherical geometry. By choosing $\alpha = 4$ and writing problem-specific code defining cell interface areas and volumes, the user can define other useful coordinate systems such as elliptical coordinates or various nozzle geometries.

Figure 3.1 shows a one-dimensional geometry in which the fluid is constrained to move along a tube. The one dimensionality is based on the assumption that the fluid variables vary very little in the direction perpendicular to the axis of the tube. The variable r measures distance along the tube. The velocity v^f is the fluid velocity along r. The points at the interfaces between cells are the finite-difference grid points. The interface positions at the beginning of a numerical timestep are denoted by $\{r^o_{i+\frac{1}{2}}\}$, where $i = 0,\ 1,\ \ldots,\ N$. At the end of a timestep Δt, the interfaces are at $\{r^n_{i+\frac{1}{2}}\}$, where

$$r^n_{i+\frac{1}{2}} \ = \ r^o_{i+\frac{1}{2}} + v^g_{i+\frac{1}{2}}\ \Delta t\ , \tag{3.1}$$

The quantities $\{v^g_{i+\frac{1}{2}}\}$ are the grid velocities, the average velocities of the cell interfaces during the interval Δt. Figure 3.1 also indicates the basic cell volumes $\{\Lambda_i\}$, and the interface areas, $\{A_{i+\frac{1}{2}}\}$. The interface areas are assumed to be perpendicular to the tube and hence to the velocities $\{v^f_{i+\frac{1}{2}}\}$. The change in the total amount of a convected quantity in a cell is the algebraic sum of the fluxes of that quantity into and out of the cell through the interfaces. Both the cell volumes $\{\Lambda_i\}$ and the interface areas $\{A_{i+\frac{1}{2}}\}$ that bound the cells have to be calculated consistently using new and old grid positions.

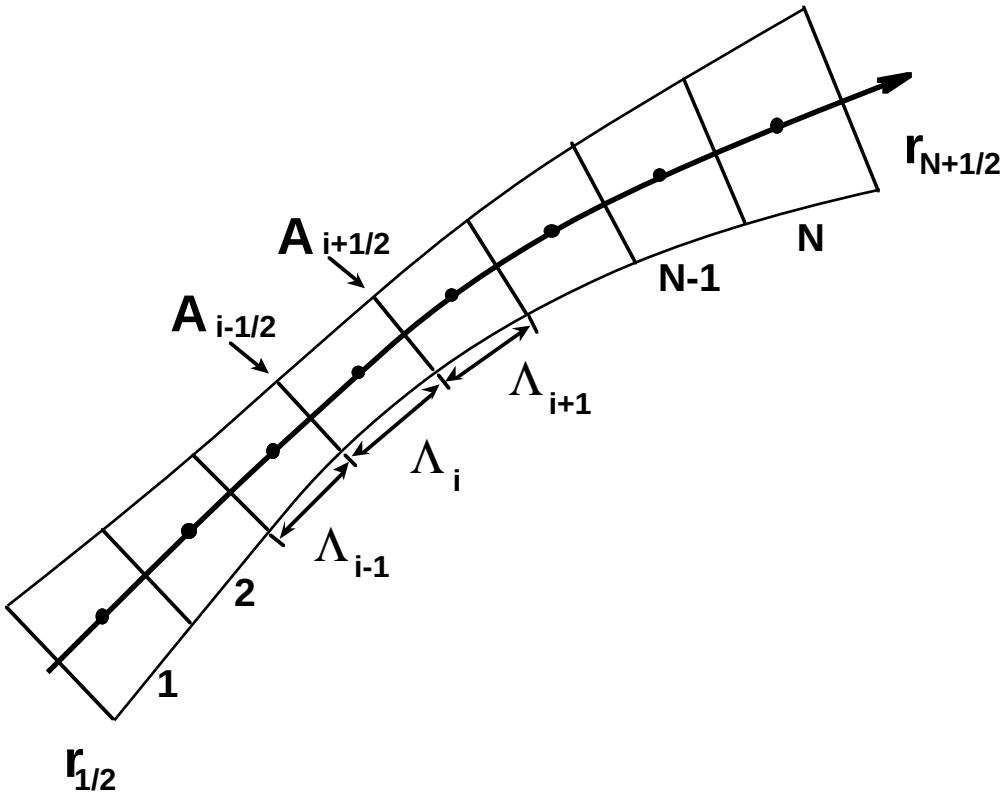

Figure 3.1 Geometry and layout of the LCPFCT finite volume grid. Physical variables are specified as cell averages in the volumes defined by the locations of the interfaces between cells and the variation of the cross-sectional area with distance between the interfaces.

The positions of the cell centers are denoted $\{r_i^{o,n}\}$ and may be related to the cell interface locations by

$$r_i^{o,n} \;=\; \frac{1}{2}\,[r_{i+\frac{1}{2}}^{o,n} + r_{i-\frac{1}{2}}^{o,n}]\;, \quad i \;=\; 1,\; 2\;, ...,\; N\;. \tag{3.2}$$

The superscripts o or n indicates the old and new grid at the beginning and the end of the timestep. The cell centers could also be computed as some weighted average of the interface locations. These locations, r_i^n and r_i^o are not needed for gasdynamics but may be useful for calculating diffusion terms added to Eq. (1.1). The boundary interface positions, $r_{\frac{1}{2}}^{o,n}$ and $r_{N+\frac{1}{2}}^{o,n}$, have to be specified by the user. For example, they might be the location of bounding walls. Then by programming $r_{\frac{1}{2}}$ as a function of time and forcing the adjacent grid points to move correspondingly, one can simulate the effect of a piston or flexible container.

To calculate convective transport, the first term on the right hand side of Eq. (1.1), we need the flux of fluid through each interface as it moves from $r_{i+\frac{1}{2}}^o$ to $r_{i+\frac{1}{2}}^n$ during a timestep. The velocities of the fluid are assumed known at the cell centers and the velocity

of the fluid at the interfaces is given by

$$v^f_{i+\frac{1}{2}} \;=\; \frac{1}{2}(v^f_{i+1} + v^f_i)\;, \qquad i \;=\; 1,\; 2,\; ...,\; N-1\;. \tag{3.3}$$

Again, other weighted averages are possible but this choice works well for all three geometries.

Because the fluxes out of one cell into the next are needed on the interfaces, we define

$$\Delta v_{i+\frac{1}{2}} \;=\; v^f_{i+\frac{1}{2}} - v^g_{i+\frac{1}{2}}, \qquad i \;=\; 1,\; 2,\; ...,\; N-1\;. \tag{3.4}$$

The boundary interface fluid velocities $\Delta v_{\frac{1}{2}}$ and $\Delta v_{N+\frac{1}{2}}$ are calculated using the locations, $r^{o,n}_{\frac{1}{2}}$ and $r^{o,n}_{N+\frac{1}{2}}$, and the two endpoint velocities $v^f_{\frac{1}{2}}$ and $v^f_{N+\frac{1}{2}}$. These velocities must also be specified as part of the problem definition because they require information from beyond the computational domain. They become part of the user-specified boundary conditions. Then

$$\begin{aligned} \Delta v_{\frac{1}{2}} \;&=\; v^f_{\frac{1}{2}} - \frac{r^n_{\frac{1}{2}} - r^o_{\frac{1}{2}}}{\Delta t}\;, \\ \Delta v_{N+\frac{1}{2}} \;&=\; v^f_{N+\frac{1}{2}} - \frac{r^n_{N+\frac{1}{2}} - r^o_{N+\frac{1}{2}}}{\Delta t}\;. \end{aligned} \tag{3.5}$$

To determine the flux on the cell boundaries, we also need the density at the cell interfaces. This is taken as

$$\rho^o_{i+\frac{1}{2}} \;=\; \frac{1}{2}\;[\rho^o_{i+1} + \rho^o_i]\;, \quad i \;=\; 1,\; 2\;, ...,\; N-1\;. \tag{3.6}$$

Weighted averages other than the simple one expressed in Eq. (3.6) are possible for this definition. The formulas

$$\begin{aligned} \rho_o \;&=\; S_1\rho_1 \;+ V_1, \\ \rho_{N+1} \;&=\; S_N\rho_N \;+ V_N, \end{aligned} \tag{3.7a}$$

are used to calculate densities at fictitious *guard* cells, here indexed 0 and $N+1$, which are imagined to exist beyond the computational domain. The quantities S_1 and S_N are slope multiplicative factors used to specify the multiplier of the value just inside the boundary to be used in the guard cells. The quantities V_1 and V_N are user specified additive values to augment the portion of the guard cell variable determined from the cell just inside the computational domain. The use of uppercase V here should not be confused with the use of a lowercase v elsewhere to denote the fluid velocity. The letters S and V are prefixed to the corresponding variable names in the computer programs to denote the boundary condition terms for the corresponding guard cell variables.

For specifying periodic boundary conditions, a logical argument in the calling sequence to LCPFCT, P_{BC}, is set to *.true.*. This corresponds to the guard cell definitions

$$\begin{aligned} \rho_o &= \rho_N \ , \\ \rho_{N+1} &= \rho_1 \ . \end{aligned} \tag{3.7b}$$

The variable P_{BC} must be *.false.* for all other cases. Both S's and V's are ignored when periodic boundaries are selected. Section 6 contains a more complete discussion of how the boundary conditions are implemented. These formulas are specified in this way because they have to be re-evaluated several times using the updated ρ values during the several stages of FCT. Thus $\rho_1 - \rho_o$ and $\rho_{N+1} - \rho_N$ are always defined at the first and last interfaces through all stages of the FCT proceedure, and Eq. (3.7) gives

$$\begin{aligned} \rho^o_{\frac{1}{2}} &= \left(\frac{1}{2} + \frac{1}{2}S_1\right)\rho^o_1 + \frac{1}{2}V_1 \ , \\ \rho^o_{N+\frac{1}{2}} &= \left(\frac{1}{2} + \frac{1}{2}S_N\right)\rho^o_N + \frac{1}{2}V_N \ . \end{aligned} \tag{3.8a}$$

or for periodic boundary conditions

$$\begin{aligned} \rho^o_{\frac{1}{2}} &= \frac{1}{2}(\rho^o_1 + \rho^o_N \ ), \\ \rho^o_{N+\frac{1}{2}} &= \frac{1}{2}(\rho^o_N + \rho^o_1 \ ). \end{aligned} \tag{3.8b}$$

Using these definitions, the convective transport part of the continuity equation is written as

$$\begin{aligned} \Lambda^o_i \rho^*_i \ = \Lambda^o_i \rho^o_i - \Delta t \ \rho^o_{i+\frac{1}{2}} \ A_{i+\frac{1}{2}} \ \Delta v_{i+\frac{1}{2}} + \Delta t \ \rho^o_{i-\frac{1}{2}} \ A_{i-\frac{1}{2}} \ \Delta v_{i-\frac{1}{2}} \ , \\ i \ = \ 1, \ 2, \ ..., \ N \ . \end{aligned} \tag{3.9}$$

The left side, $\Lambda^o_i \rho^*_i$, has not yet undergone the compression or expansion that changes Λ^o_i to Λ^n_i . The source terms have not yet been incorporated and the diffusion and antidiffusion portions of flux correction still have to be included.

The source terms in Eq. (1.1) are added into Eq. (3.9),

$$\begin{aligned} \Lambda^o_i \rho^T_i \ &= \Lambda^o_i \rho^*_i + \frac{1}{2}\Delta t \ A_{i+\frac{1}{2}}(D_{1,i+1} + D_{1,i}) - \frac{1}{2}\Delta t \ A_{i-\frac{1}{2}}(D_{1,i} + D_{1,i-1}) \\ &+ \frac{1}{4}\Delta t \ C_{2,i}(A_{i+\frac{1}{2}} + A_{i-\frac{1}{2}}) \ (D_{2,i+1} - D_{2,i-1}) \\ &+ \Delta t \ \Lambda^o_i D_{3,i} \ , \qquad i \ = \ 2, \ ..., \ N-1 \ . \end{aligned} \tag{3.10}$$

The end values, at cells $i = i1$ and $i = iN$, are computed using $D_{k,I1-\frac{1}{2}}$ and $D_{k,IN+\frac{1}{2}}$, the first and last interface values of D, which must be specifically specified by the user, in place of the interface average at the boundaries. Other source terms can be added easily to the formalism, but the three source terms in Eq. (1.1) are adequate to treat most important applications.

The diffusion stage of this FCT algorithm also includes the cell volume change when the grid is moving,

$$\begin{aligned} \Lambda_i^n \tilde{\rho}_i \ &= \Lambda_i^o \rho_i^T + \nu_{i+\frac{1}{2}} \Lambda_{i+\frac{1}{2}} (\rho_{i+1}^o - \rho_i^o) \\ &\quad - \nu_{i-\frac{1}{2}} \Lambda_{i-\frac{1}{2}} (\rho_i^o - \rho_{i-1}^o) \ , \quad i \ = \ 1, \ 2, \ ..., \ N \ . \end{aligned} \tag{3.11}$$

The quantities $\{\tilde{\rho}_i\}$ make up the transported-diffused density profile. The diffusion coefficients can be chosen to reduce phase errors from second to fourth order. The interface-averaged volumes $\{\Lambda_{i+\frac{1}{2}}\}$ multiply the $\{\nu_{i+\frac{1}{2}}\}$ in Eq. (3.11) and are defined as

$$\Lambda_{i+\frac{1}{2}} \ = \ \frac{1}{2}(\Lambda_{i+1}^n + \Lambda_i^n) \ , \quad i \ = \ 1, \ 2, \ ..., \ N-1 \ . \tag{3.12}$$

The boundary interface volumes are chosen as

$$\begin{aligned} \Lambda_{\frac{1}{2}} \ &= \ \Lambda_1^n \ , \\ \Lambda_{N+\frac{1}{2}} \ &= \ \Lambda_N^n \ . \end{aligned} \tag{3.13}$$

The convection, additional source terms, compression, and diffusion have been broken into the successive stages shown in Eqs. (3.9), (3.10), and (3.11) because we need to compute the antidiffusive fluxes using $\{\rho_i^T\}$. If the antidiffusive flux is computed using $\{\tilde{\rho}_i\}$, that is, after the diffusion has been added, the algorithm has residual diffusion both when the grid is Lagrangian, $v^f = v^g$, and in the special case when both the grid and the fluid are stationary. Therefore the transported but not diffused values, $\{\rho_i^T\}$ are used to calculate the raw, uncorrected antidiffusive fluxes,

$$f_{i+\frac{1}{2}}^{ad} \ = \ \mu_{i+\frac{1}{2}} \Lambda_{i+\frac{1}{2}} [\rho_{i+1}^T - \rho_i^T] \ , \quad i \ = \ 0, \ 1, \ ..., \ N \ . \tag{3.14}$$

The antidiffusion is designed so that when the grid is Lagrangian and $\{\Delta v_{i+\frac{1}{2}}\}$ vanishes in Eq. (3.9),

$$\Lambda_i^n \rho_i^n \ = \ \Lambda_i^o \rho_i^o \ . \tag{3.15}$$

Substituting Eq. (3.9) and (3.10) into Eq. (3.11) in the Langrangian case with no sources gives

$$\Lambda_i^n \tilde{\rho}_i \ = \ \Lambda_i^o \rho_i^o + \nu_{i+\frac{1}{2}} \Lambda_{i+\frac{1}{2}} (\rho_{i+1}^o - \rho_i^o) - \nu_{i-\frac{1}{2}} \Lambda_{i-\frac{1}{2}} (\rho_i^o - \rho_{i-1}^o) \ , \tag{3.16}$$

because $\rho_i^T = \rho_i^o$. The antidiffusion procedure, applied to Eq. (3.16), gives

$$\begin{aligned} \Lambda_i^n \rho_i^n \;&= \Lambda_i^o \rho_i^o + (\nu_{i+\frac{1}{2}} - \mu_{i+\frac{1}{2}}) \Lambda_{i+\frac{1}{2}} (\rho_{i+1}^o - \rho_i^o) \\ &\quad - (\nu_{i-\frac{1}{2}} - \mu_{i-\frac{1}{2}}) \Lambda_{i-\frac{1}{2}} (\rho_i^o - \rho_{i-1}^o) \; . \end{aligned} \tag{3.17}$$

When the grid is Lagrangian, the desired result of Eq. (3.15) can be achieved as long as

$$\nu_{i+\frac{1}{2}} = \mu_{i+\frac{1}{2}} \; . \tag{3.18}$$

Boris and Book (1976) explain that the choices

$$\begin{aligned} \nu_{i+\frac{1}{2}} \;&\equiv\; \frac{1}{6} + \frac{1}{3} \, \epsilon_{i+\frac{1}{2}}^2 \; , \\ \mu_{i+\frac{1}{2}} \;&\equiv\; \frac{1}{6} - \frac{1}{6} \, \epsilon_{i+\frac{1}{2}}^2 \; , \end{aligned} \tag{3.19}$$

reduce the relative phase errors in convection on a locally uniform grid to fourth order. By defining

$$\epsilon_{i+\frac{1}{2}} \;\equiv\; A_{i+\frac{1}{2}} \Delta v_{i+\frac{1}{2}} \; \frac{\Delta t}{2} \left[\frac{1}{\Lambda_i^n} + \frac{1}{\Lambda_{i+1}^n} \right] \; , \quad i \;=\; 0, \; 1, \; ..., \; N \; , \tag{3.20}$$

the diffusion and antidiffusion coefficients are automatically equal in the Lagrangian case. Then Eqs. (3.19) are satisfied for the portion of the fluid motion that convects material through the moving interfaces.

As in Eq. (2.27) above, the signed quantites $\{S_{i+\frac{1}{2}}\}$ can be defined with the sign of $[\tilde{\rho}_{i+1} - \tilde{\rho}_i]$ and magnitude unity. Using $\{f_{i+\frac{1}{2}}^{ad}\}$ from Eq. (3.14) as the raw antidiffusive fluxes and $\{\tilde{\rho}_i\}$ from Eq. (3.11), the corrected antidiffusive flux is

$$\begin{aligned} f_{i+\frac{1}{2}}^c \;&=\; S_{i+\frac{1}{2}} \; \max \Big\{ 0, \; \min \left[|f_{i+\frac{1}{2}}^{ad}|, \; S_{i+\frac{1}{2}} \Lambda_{i+1}^n (\tilde{\rho}_{i+2} - \tilde{\rho}_{i+1}), \right. \\ &\qquad \left. S_{i+\frac{1}{2}} \Lambda_i^n (\tilde{\rho}_i - \tilde{\rho}_{i-1}) \right] \Big\} \; , \quad i \;=\; 1, 2, ..., N-1 \; . \end{aligned} \tag{3.21}$$

For correcting the boundary fluxes $f_{\frac{1}{2}}^c$ and $f_{N+\frac{1}{2}}^c$, the min[..., ..., ...] term in Eq. (3.21) contains only two terms. The correction coming from a difference reaching beyond the boundary is simply dropped from the calculation except for periodic boundaries where a periodic application of the differences is used. The result, $\{\rho_i^n\}$, is then computed as in Eq. (2.26), where the corrected fluxes $\{f_{i+\frac{1}{2}}^c\}$ replace $\{f_{i+\frac{1}{2}}^{ad}\}$. The final density at the new time is

$$\rho_i^n \;=\; \tilde{\rho}_i - \frac{1}{\Lambda_i^n} \left[f_{i+\frac{1}{2}}^c - f_{i-\frac{1}{2}}^c \right] \; . \tag{3.22}$$

A few of the geometric variables used above have yet to be defined. The obvious choice of volume elements, at the beginning and end of the timesteps, in Cartesian, cylindrical, and spherical geometries are

$$\Lambda_i^{o,n} = \begin{cases} [r_{i+\frac{1}{2}}^{o,n} - r_{i-\frac{1}{2}}^{o,n}], & \text{Cartesian} \\ \pi[(r_{i+\frac{1}{2}}^{o,n})^2 - (r_{i-\frac{1}{2}}^{o,n})^2] & \text{cylindrical} \\ \frac{4}{3}\pi[(r_{i+\frac{1}{2}}^{o,n})^3 - (r_{i-\frac{1}{2}}^{o,n})^3] & \text{spherical .} \end{cases} \tag{3.23}$$

The corresponding interface areas are

$$A_{i+\frac{1}{2}} = \begin{cases} 1 & \text{Cartesian} \\ \pi[r_{i+\frac{1}{2}}^{o} + r_{i+\frac{1}{2}}^{n}] & \text{cylindrical} \\ \frac{4}{3}\pi[(r_{i+\frac{1}{2}}^{o})^2 + r_{i+\frac{1}{2}}^{o} r_{i+\frac{1}{2}}^{n} + (r_{i+\frac{1}{2}}^{n})^2] & \text{spherical .} \end{cases} \tag{3.24}$$

The interface areas are time and space centered. Though other centered choices are also possible, these particular definitions ensure that a constant density ρ remains constant and unchanged when the fluid is at rest but the grid is rezoned arbitrarily. Depending on how the LCPFCT boundary condition factors are chosen, a subject considered in both Sections 5 and 6, the time-variable grid can even move fluid into and out of the system while the density while a constant density remains constant.

4. SPLIT STEP APPLICATION OF MONOTONE FCT ALGORITHMS

Section 3 described the monotone FCT algorithm for integrating a single continuity equation using LCPFCT. We now extend the approach to solving coupled continuity equations. Specifically, we want to solve the three conservative continuity equations of gas dynamics simultaneously,

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot \rho \mathbf{v}, \tag{4.1}$$

$$\frac{\partial \rho \mathbf{v}}{\partial t} = -\nabla \cdot (\rho \mathbf{v}\mathbf{v}) - \nabla P \tag{4.2}$$

and

$$\frac{\partial E}{\partial t} = -\nabla \cdot E\mathbf{v} - \nabla \cdot (\mathbf{v}P) \ . \tag{4.3}$$

First, we consider this problem in one spatial dimension using a two stage Runge-Kutta time integration. Then split step procedures are introduced to combine several one-dimensional calculations to create a multidimensional monotone calculation. Section 5 expands the discusssion of this section to the practical aspects of using the LCPFCT routines to carry out the general procedures described here. LCPFCT can be used to solve systems of continuity equations for many applications but compressible gas dynamics is the most widespread use and serves as an ideal example to illustrate the various necessary steps and techniques.

4.1 One-Dimensional Solution of Coupled Continuity Equations

Solving the coupled equations (4.1–4.3) is best done by determining the timestep, then integrating from the old time t^o forward a half timestep to $t^o + \frac{\Delta t}{2}$, and then integrating from t^o to the full timestep $t^o + \Delta t$. The results of the half-step integration are used to evaluate time-centered spatial derivatives and fluxes. Assume that the cell-averaged values of all fluid quantities are known at t^o. The integration procedure for one timestep is:

1. Integrate the equations for a half timestep to find first-order accurate approximations to the fluid variables at the middle of the timestep ("time-centered"). This requires one to:

   a. Calculate $\{v_i^o\}$ and $\{P_i^o\}$ using the old values of $\{\rho_i^o\}$, $\{\rho_i^o v_i^o\}$, and $\{E_i^o\}$ known at the beginning of the timestep.

   b. Convect $\{\rho_i^o\}$ a half timestep to $\{\rho_i^{\frac{1}{2}}\}$. (Here the superscript $\frac{1}{2}$ is used to indicate a variable at the new half timestep, not the square root).

   c. Evaluate $-\nabla P^o$ as the source term for the momentum equation.

d. Convect $\{\rho_i^o v_i^o\}$ to $\{\rho_i^{\frac{1}{2}} v_i^{\frac{1}{2}}\}$ using $-\nabla P^o$.

e. Evaluate $-\nabla \cdot (P^o v^o)$ as the source term for the energy equation.

f. Convect $\{E_i^o\}$ for a half timstep $\frac{\Delta t}{2}$ to $\{E_i^{\frac{1}{2}}\}$ using $-\nabla \cdot (P^o v^o)$.

2. Integrate the equations for a whole timestep to find results which are second-order accurate in time at the end of the timestep $t^o + \Delta t$.

a. Calculate $\{v_i^{\frac{1}{2}}\}$ and $\{P_i^{\frac{1}{2}}\}$ using the half-step values $\{\rho_i^{\frac{1}{2}}\}$, $\{\rho_i^{\frac{1}{2}} v_i^{\frac{1}{2}}\}$, and $\{E_i^{\frac{1}{2}}\}$.

b. Convect $\{\rho_i^o\}$ for the full timestep Δt to $\{\rho_i^1\}$.

c. Evaluate $-\nabla P^{\frac{1}{2}}$ for the momentum sources.

d. Convect $\{\rho_i^o v_i^o\}$ to $\{\rho_i^1 v_i^1\}$ using $-\nabla P^{\frac{1}{2}}$.

e. Evaluate $-\nabla \cdot P^{\frac{1}{2}} v^{\frac{1}{2}}$ for the energy sources.

f. Convect $\{E_i^o\}$ to $\{E_i^1\}$ using $-\nabla \cdot P^{\frac{1}{2}} v^{\frac{1}{2}}$.

3. Repeat these two procedures above to do another timestep from t^1 to t^2.

This two-step, second-order time integration increases the accuracy of the calculations significantly.

Often we want to couple N_s chemical species equations to Eqs. (4.1) – (4.3),

$$\frac{\partial n_s}{\partial t} \;=\; -\nabla \cdot n_s \mathbf{v} \;, \qquad s \;=\; 1, ..., N_s \tag{4.4}$$

where $n_s(r,t)$ is the number density of species s and the subscript s is used here to avoid confusion with i, generally used above as a cell or interface index. In general you do not have to split the timestep for these variables provided that the half-step velocities are used in advancing $\{n_i^o\}$ to $\{n_i^1\}$. After integrating the fluid variables for the half and whole timestep, convect these species the full timestep using the centered velocities $\{v_i^{\frac{1}{2}}\}$. If the half-step values of these variables affect either $\{v_i^{\frac{1}{2}}\}$ or $\{P_i^{\frac{1}{2}}\}$, the half-step integration for the $\{n_i\}$ would also have to be performed.

4.2 Multidimensions through Timestep Splitting

One-dimensional continuity equation solvers such as LCPFCT can be used repetitively to construct a multidimensional program by timestep splitting in the different coordinate directions. This approach is straightforward when an orthogonal grid can be constructed with physical boundaries along segments of grid lines. Various geometries, such as $(x-y)$, $(r-z)$, or in general orthogonal coordinates $(\eta-\xi)$, can be integrated by timestep splitting.

The approach can also be extended to three dimensions and to fully general geometries with the addition of special boundary algorithms taking into account the variation of cell areas and volumes when a general curved boundary intersects the regular orthogonal grid.

For example, the four equations which describe ideal two-dimensional gas dynamics in Cartesian $(x-y)$ geometry are:

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_y) - \frac{\partial}{\partial x}(\rho v_x) \\
\frac{\partial \rho v_x}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_x v_y) - \frac{\partial}{\partial x}(\rho v_x v_x) - \frac{\partial P}{\partial x} \\
\frac{\partial \rho v_y}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_y v_y) - \frac{\partial}{\partial x}(\rho v_y v_x) - \frac{\partial P}{\partial y} \\
\frac{\partial E}{\partial t} &= -\frac{\partial}{\partial y}\left[(E+P)v_y\right] - \frac{\partial}{\partial x}\left[(E+P)v_x\right] .
\end{aligned} \tag{4.5}$$

The pressure and energy are related by

$$E = \epsilon + \frac{1}{2}\rho(v_x^2 + v_y^2) . \tag{4.6}$$

where $\epsilon \equiv P/(\gamma - 1)$. The right sides of Eqs. (4.5) are separated into two parts, the y-direction terms and the x-direction terms. This arrangement in each of the four equations separates the y-derivatives and the x-derivatives in the divergence and gradient terms into parts which can be treated sequentially by a general one-dimensional continuity equation solver.

Each y-direction column in the grid is integrated using the one-dimensional LCPFCT module to solve the four coupled continuity equations (4.5) from time t to $t + \Delta t$. The

y-direction split-step equations to be solved are

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_y) \\
\frac{\partial \rho v_x}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_x v_y) \\
\frac{\partial \rho v_y}{\partial t} &= -\frac{\partial}{\partial y}(\rho v_y v_y) - \frac{\partial P}{\partial y} \\
\frac{\partial E}{\partial t} &= -\frac{\partial}{\partial y}(E v_y) - \frac{\partial}{\partial y}(P v_y) \ .
\end{aligned} \tag{4.7}$$

Equations (4.7) are in the form of the general continuity equation (1.1) with $\alpha = 1$ for planar geometry. Because the y gradients and fluxes are being treated together, the one-dimensional integration connects those cells which are influencing each other through the y-component of convection.

The changes due to the derivatives in the x-direction must now be included. This is done in a second split step of one-dimensional integrations along each x-column,

$$\begin{aligned}
\frac{\partial \rho}{\partial t} &= -\frac{\partial}{\partial x}(\rho v_x) \\
\frac{\partial \rho v_x}{\partial t} &= -\frac{\partial}{\partial x}(\rho v_x v_x) - \frac{\partial P}{\partial x} \\
\frac{\partial \rho v_y}{\partial t} &= -\frac{\partial}{\partial x}(\rho v_y v_x) \\
\frac{\partial E}{\partial t} &= -\frac{\partial}{\partial x}(E v_x) - \frac{\partial}{\partial x}(P v_x)
\end{aligned} \tag{4.8}$$

where $\alpha = 1$ in Eq. (1.1) for planar geometry. The x and y integrations are alternated, each pair of sequential integrations constituting a full convection timestep. Thus a single optimized algorithm for a reasonably general continuity equation can be used to build up multidimensional fluid dynamics models. Analogous equations for axisymmetric geometry have been written out in Oran and Boris (1987).

To use this split-step approach, the timestep must be small enough that the distinct components of the fluxes do not change the cell-averaged values appreciably during the

timestep. This approach is second-order accurate as long as the timestep is small and changed slowly enough, but there is still a bias built in depending on which direction, x or y, is integrated first. To remove this bias, the results from two calculations for each timestep can be averaged, an expensive but effective solution. Alternately a fully multidimensional FCT algorithm can be used such as developed by Zalesak (1979, 1981) or DeVore (1989,1991) with some corresponding extra cost and complication. Generally this sequencing bias is quite small and is usually ignored.

5. HOW TO USE LCPFCT

The set of Fortran subroutines that make up the LCPFCT library is listed in Appendix A. This library contains a main subroutine, called LCPFCT, and several auxiliary subroutines. This structure serves both efficiency and flexibility: minimizing the need to repeat common calculations and providing flexibility in defining various geometries, source terms, and boundary conditions. Thus, for example, a single velocity profile may be used to convect a number of different continuity equations representing different chemical species, fluid phases, or ionization states. All velocity-dependent coefficients that are common to the FCT algorithms for these several equations during a particular timestep and integration direction are computed once by subroutine VELOCITY and placed in a common block for use by the repeated calls to LCPFCT which will integrate each of the equations separately. An entire calculation, therefore, requires a sequence of calls 1) to define the geometry by specifying a suitable computational grid, 2) to calculate a number of velocity-dependent factors, 3) to establish the boundary condition coefficients for the next continuity equation to be integrated, 4) to calculate the source terms in Eqs. (4.1–4.3), and 5) to advance the fluid variables one continuity equation at a time.

5.1 LCPFCT Variables in Common

Information is passed between the user's program and the LCPFCT library generally through the arguments to the subroutine calls. Information is passed between the various subroutines of the LCPFCT library both through the subroutine arguments and through named common blocks. The user is asked to control and store the information pertaining specifically to his problem and the particular continuity equations being solved while the LCPFCT library controls all data that might be reused or that is particular to the FCT algorithms being used to integrate and manage the grid and the equations.

Each of the subroutines in the library performs a different, well-defined task. The main subroutine LCPFCT convects the variables, and it must be called for each variable integrated at each timestep or partial timestep. Subroutine MAKEGRID calculates the geometric coefficients and must be called any time the grid cell interfaces are changed. The subroutine VELOCITY updates the velocity-dependent coefficients and must be called each time the convective velocity or the grid is changed. Subroutine SOURCES is called separately each time a new source term for a continuity equation needs to be evaluated. Thus none, one, or several different source terms may be added together and passed to the continuity equation solver. LCPFCT then resets the source terms to zero at the end of its execution. Thus source terms, which generally are not reuseable in any case, must be recomputed for each call to LCPFCT. Table 5.1 below defines the variables in FCT_GRID and their equivalent algorithmic/physical definition is given in Section 3. These library common blocks generally begin with the prefix FCT_.

Table 5.1 Variables in LCPFCT Common Blocks

	Type	Text Symbol	Meaning
FCT_GRID variables			
ROH	RA(N+1)*	$r^o_{i-\frac{1}{2}}$	Cell boundary location; start of timestep
RNH	RA(N+1)	$r^n_{i-\frac{1}{2}}$	Cell boundary location; end of timestep
LO	RA(N)	Λ^o_i	Cell volume; start of timestep
LN	RA(N)	Λ^n_i	Cell volume; end of timestep
AH	RA(N+1)	$A_{i-\frac{1}{2}}$	Cell interface area; average over timestep
LH	RA(N+1)	$\Lambda_{i-\frac{1}{2}} = \frac{1}{2}[\Lambda^n_i + \Lambda^n_{i-1}]$	
RLN	RA(N)	$1/\Lambda^n_i$	
RLH	RA(N+1)	$\frac{1}{2}[1/\Lambda^n_i + 1/\Lambda^n_{i-1}]$	
DIFF	RA(N+1)		Array used for grid differences in MAKEGRID
FCT_VELO variables			
HADUDTH	RA(N+1)	$\frac{1}{2}\Delta t A_{i-\frac{1}{2}} \Delta v_{i-\frac{1}{2}}$	Interface flux coefficient to determine transported flux
EPSH	RA(N+1)	$\Delta v_{i-\frac{1}{2}} \Delta t \ A_{i-\frac{1}{2}}/\Lambda_{i-\frac{1}{2}}$	Nondimensional interface velocity
NULH	RA(N+1)	$\nu_{i-\frac{1}{2}} \Lambda_{i-\frac{1}{2}}$	Diffusion flux coefficient
MULH	RA(N+1)	$\mu_{i-\frac{1}{2}} \Lambda_{i-\frac{1}{2}}$	Raw antidiffusion flux coefficient
ADUGTH	RA(N+1)	$A_{i-\frac{1}{2}}[r^n_{i-\frac{1}{2}} - r^o_{i-\frac{1}{2}}]$	Volume swept out by interface
VDTODR	RA(N+1)	$\frac{\Delta t \Delta v_{i-\frac{1}{2}}}{[r^n_{i+\frac{1}{2}} - r^n_{i-\frac{3}{2}}]}$	Nonconservative form of transport coefficient
FCT_MISC variables			
SOURCE	RA(N)	$\{\ \}^\dagger$	Sum of all the source terms
DIFF1	REAL		Residual diffusion; value of $S_{i-\frac{1}{2}}$ used in antidiffusion coefficient; default = 1.000
FCT_NDEX variables			
SCALARS	RA(NIND)	$\{\ \}^\dagger$	Additive scalar source terms
INDEX	IA(NIND)		Scalar list of cells to receive added sources
NIND	INTEGER		Number of nonzero scalars in the indexed list

Table 5.1 Variables in LCPFCT Common Blocks **(Cont.)**

	Type	Text Symbol	Meaning
FCT_SCRH variables			
LNRHOT	RA(N)*	$\Lambda_i^n \bar{\rho}_i$ and $\Lambda_i^n \rho_i^n$	Transported/diffused mass elements
LORHOT	RA(N)	$\Lambda_i^o \rho_i^*$ and $\Lambda_i^o \rho_i^T$	Transported mass elements
FSGN	RA(N+1)	$S_{i-\frac{1}{2}}$	Sign of grid differences of $\bar{\rho}_i$
FABS	RA(N+1)	$\lvert f_{i-\frac{1}{2}}^{ad} \rvert$	Absolute value of raw antidiffusive flux
FLXH	RA(N+1)	multiple uses	Used for convective and diffusive fluxes
TERP	RA(N+1)	$S_{i-\frac{1}{2}} \Lambda_i^n (\bar{\rho}_{i+1} - \bar{\rho}_i)$	Antidiffusive flux limit from right
TERM	RA(N+1)	$S_{i-\frac{3}{2}} \Lambda_{i-1}^n (\bar{\rho}_{i-1} - \bar{\rho}_{i-2})$	Antidiffusive flux limit from left
RHOT	RA(N)	ρ_i^T	Transported, sources added and compressed density
RHOTD	RA(N)	$\tilde{\rho}_i$	Transported diffused, sources added and compressed density
SCRH	RA(N)	multiple uses	Source term scratch space
SCR1	RA(N)	multiple uses	Source term scratch space

* RA(N) stands for Real Array of length N.

** IA(N) stands for Integer Array of length N.

† Source terms defined in Eq. (3.10)

As will be seen in the appendices, all of the one-dimensional LCPFCT arrays meant to span the maximum size of the physical grid are dimensioned NPT (number of points) and this is set to 202 everywhere by a parameter statement. When a bigger problem is attempted, for example a 250×150 grid in two dimensions, this size should be increased. By convention we leave two more points than the maximum line through the system although one extra should be enough.

The LCPFCT library is structured so that information that needs to be passed from the user's control program to LCPFCT is done through arguments in various subroutine calls. The user generally does not need to access the internal variables stored in the named common blocks. There are exceptions to this, however, when a more advanced user may want to and can access these internal variables. For example, the common block FCT_GRID stores and transmits information about the grid locations, grid motion, cell volumes and cell interface areas. One can include FCT_GRID in a user-supplied subroutine where the volumes and areas of the cells are computed for a nonstandard user-specified grid

geometry. Another example could involve user modification of the antidiffusion coefficients (Fortran variable $MULH$) in FCT_VELO to steepen known fluid interfaces or contact surfaces.

FCT_VELO contains velocity-dependent information about diffusion and antidiffusion coefficients and fluxes through the cell interfaces. FCT_MISC includess the array containing the accumulated source terms and the residual diffusion coefficient, $DIFF1$ (defaulted to 0.999). $DIFF1$ may be reset by a call to subroutine RESIDIFF to unity for minimal residual diffusion or to slightly smaller values than 0.999. FCT_SCRH contains a number of internal scratch vectors used by the LCPFCT subroutine and other subroutines of the libary. They are in common to allow reuse of the space but no information, by convention, is ever passed between subroutines using this common block. Equivalencing the variables in FCT_SCRH with scratch storage in the user's program provides a way to save space but the capacity of modern computers generally makes this a needless economy. There is also a common block FCT_NDEX which appears only in SOURCES, ZERODIFF, AND ZEROFLUX. FCT_NDEX is intended to convey scalar indexed source terms directly from a user program to SOURCES, or indexed lists of cells to ZERODIFF or ZEROFLUX, without need for a more complex calling sequence. These and other miscellaneous uses of the LCPFCT library including a few special auxiliary routines are discussed in Section 7.

The LCPFCT subroutines, their arguments, and their calling sequences are decribed in detail in the subsections below, both in tabular form and in explanatory text. The Fortran subroutines themselves are reproduced in Appendix A. There are, in addition several other library subroutines in Appendix A whose use is often not necessary or which are needed only for some simple auxiliary tasks (ZERODIFF, ZEROFLUX, CONSERVE) or for special types of applications (COPYGRID, NEW_GRID, SET_GRID). These are discussed only briefly here and in Section 7 on "Esoterica" but they also appear in Appendix A. By reading through the LCPFCT subroutine programs in Appendix A, the reader can determine the function of each routine exactly by looking at the quantities which each uses, computes, and leaves in the named common blocks.

5.2 Subroutines in LCPFCT

Subroutine MAKEGRID: –

Subroutine MAKEGRID sets up the grid parameters and computes cell volumes and interface areas for the entire LCPFCT library. The information computed by MAKEGRID is passed through common block FCT_GRID to subroutines VELOCITY, SOURCES, CONSERVE, CNVFCT, and LCPFCT. Table 5.2 shows the calling sequence and defines the arguments for MAKEGRID. The single subroutine MAKEGRID subsumes the functions played by the three subroutines IGRIDD, NGRIDD, and OGRIDD in the previously pub-

Table 5.2 Arguments in Subroutine MAKEGRID

CALL MAKEGRID (RADHO, RADHN, I1, INP, ALPHA)			
Variable Name	Type	Text symbol	Meaning
RADHO	RA(INP)*	$r^o_{i-\frac{1}{2}}$	Cell interfaces at start of timestep
RADHN	RA(INP)*	$r^n_{i-\frac{1}{2}}$	Cell interfaces at end of timestep
I1	Integer	$I1 - \frac{1}{2}$	Index of first active cell interface
INP	Integer	$IN + \frac{1}{2}$	Index of last active cell interface
ALPHA	Integer	α	= 1 for planar (Cartesian) geometry = 2 for cylinderical geometry = 3 for spherical geometry = 4 user-supplied geometry

* RA(N) stands for Real Array of length N .

lished version of the FCT library. This is the single greatest change from a user's perspective in the current version and was made to simplify control of the grid at the cost of a few arithmetic operations which strictly speaking can be avoided in some applications. MAKEGRID is used to generate the geometry-dependent coefficients for a particular line of integration in a particular direction. If the grid is fixed and only one direction of integration is active, one call to MAKEGRID is the only geometric call needed. If all rows of a two-dimensional calculation have the same grid, MAKEGRID only need be called once before the first row integration of a timestep. Then it must be called again before beginning column integrations to complete the split multidimensional timestep. The calling sequence to subroutine MAKEGRID has five arguments:

1. **RADHO** – A real array containing the location of the "old" interfaces of the grid cells, which is referred to in Section 3 as $r^o_{i+\frac{1}{2}}$.

2. **RADHN** – A real array containing the location of the "new" interfaces of the grid cells, which is referred to in Section 3 as $r^n_{\frac{i+1}{2}}$.

3. **I1** – The first index of the old and new cell interfaces supplied in $RADHO$ and $RADHN$. Typically the grid is set up from cell and interface $I1$ across the entire system to cell IN and interface $IN + 1$ even though any specific integration may use only a portion of the grid. Care should be exercised if $I1$ does not equal 1.

4. **INP** – The last index of the old and new active cell interfaces supplied in $RADHO$

and $RADHN$.

5. **ALPHA** – The grid geometry indicator. $\alpha = 1$ for cartesian geometry, $\alpha = 2$ for cylinderical geometry, and $\alpha = 3$ for spherical geometry. The fourth option, $\alpha = 4$, requires the user to write a subroutine called before the call MAKEGRID. This subroutine must compute the timestep centered interface areas, $\{AH_{\frac{i-1}{2}}\}$, and the old and new cell volumes $\{LO_i\}$ and $\{LN_i\}$, placing the results in common block FCT_GRID. Even for the case of the user supplied geometry, the old and new interface locations are passed to MAKEGRID through the arguments RADHO and RADHN.

Subroutine VELOCITY: –

Table 5.3 Arguments in Subroutine VELOCITY

CALL VELOCITY (UH, I1, INP, DT)			
Variable name	Type	Text symbol	Meaning
UH	RA(INP)*	$v_{i-\frac{1}{2}}$	Cell interface velocities
I1	Integer	$I1 - \frac{1}{2}$	index of the first interface integrated
INP	Integer	$IN + \frac{1}{2}$	index of the last interface integrated
DT	Real	Δt	time step

* RA(N) stands for Real Array of length N .

After the grid geometric factors are computed, subroutine VELOCITY is used to compute the velocity-dependent coefficients for the convective transport, diffusion, and antidiffusion. This call must be made each time the convective transport velocity is changed. VELOCITY need be called only once if, for example, the velocity is constant in time. However, for a typical fluid problem with a second-order time integration, VELOCITY must be called twice for each row and column of the grid: once with the velocities at the start of the timestep and again after the half step using the velocities computed from the half step. VELOCITY must also be called each time there is a change in the grid locations even if the velocity field itself has not changed. The calling sequence to subroutine VELOCITY has four arguments:

1. **UH** – The fluid velocities, $\{v^{0,\frac{1}{2}}_{i-\frac{1}{2}}\}$, on the cell interfaces, $\{r_{i-\frac{1}{2}}\}$. These must be computed as some average of the cell centered velocities $\{v_i^{0,\frac{1}{2}}\}$,

2. **I1** – The interface index of the first cell, at $r^n_{I1-\frac{1}{2}}$, in the domain to be integrated. This is the location where boundary conditions at one side of the domain of integration are specified.

3. **INP** – The interface index of the last cell, at $r^n_{IN+\frac{1}{2}}$, in the domain to be integrated. This is the location where boundary conditions at the other side of the domain of integration are specified.

4. **DT** – the timestep.

The velocities $v^{0,\frac{1}{2}}_{I1-\frac{1}{2}}$ and $v^{0,\frac{1}{2}}_{IN+\frac{1}{2}}$ are also the boundary conditions on the velocity and determine the flux of the conserved quantity through the boundary. If the velocity of the grid at the boundary is equal to UH there, the convection flux of the conserved quantity through the boundary should be identically zero.

The LCPFCT routines also allow the user to integrate variables in a selected part of the grid. For example, one can eliminate a corner of a two-dimensional grid from the integration or allow this corner to be integrated seperately. Similarly the user can implement special boundary conditions in the interior of the grid. This is done by using the parameter $I1$ (different from 1) to define the first cell to be integrated or reducing $INP = IN + 1$, the last cell interface. It is important to remember that the grids and fluxes are defined on the boundaries of cells and that INP should be the value of the last cell to be integrated plus one. There should always be $IN + 1$ interfaces when IN cells are integrated.

Subroutine SOURCES: –

Once the call to VELOCITY has been made, these velocity-dependent coefficients can be used to convect several different conserved quantities by corresponding separate calls to the main subroutine LCPFCT. For the solution of the coupled fluid dynamics equations, this involves a call to LCPFCT for the mass density, each momentum density, and the energy density. For each of these equations, source terms are added by a call to the subroutine SOURCES immediately prior to the corresponding call to LCPFCT. The sum of several different sources can be accumulated by a sequence of calls to SOURCES, which keeps the running sum from all SOURCES calls since the last call to LCPFCT in the real array SOURCE in FCT_MISC. Once LCPFCT uses SOURCE, it is reset to zero so new sources can be added or it can be left at zero until it is needed again. The calling sequence of subroutine SOURCES has eight arguments:

1–2. **I1** and **IN** are the first and last cells integrated using the velocities set up by the last call to VELOCITY.

Table 5.4 Arguments in Subroutine SOURCES

Call SOURCES (I1, IN, DT, MODE, C, D, D1, DN)			
Variable Name	Type	Text symbol	Meaning
I1	Integer	$_1$	Location of first cell of integration
IN	Integer	$_N$	Location of last cell of integration
DT	REAL	Δt	Timestep
MODE	Integer		= 1 computes $\nabla \cdot D_1$ conservatively = 2 computes $C_2 \nabla D_2$ = 3 adds D_3 to the sources = 4 $\nabla \cdot D_1$ from interface data D = 5 $C_2 \nabla D_2$ from interface data D = 6 adds $+C$ for selected indexed cells
C	RA(N)*	$C_{k,i}$	Array of source variables
D	RA(N)	$D_{k,(i,i-\frac{1}{2})}$	Array of source variables
D1	Real	D_{I1}	First interface value of D (if needed)
DN	Real	D_{INP}	Last interface value of D (if needed)

* RA(N) stands for Real Array of length N .

3. **DT** is the current timestep used to advance the convected quatities. The value of DT passed to sources should be equal to one half the whole-timestep value for the half timestep integration in a fluid dynamics calculation.

4. **MODE**, an integer, determines the types of source terms included. SOURCES computes divergence, gradient, and additive source terms according to Eq. (3.10). When $MODE = 1$, the source term $(\nabla \cdot D)$ in Eq. (3.10) is computed using the cell-centered array D. When $MODE = 2$, the source term $(C \nabla D)$ in Eq. (3.10) is computed from cell-centered arrays C and D. $MODE = 3$ adds an externally computed source term D_3 as in Eq. (3.10). $MODE = 4$ and $MODE = 5$ are used to compute the same sources as $MODE = 1$ and $MODE = 2$, except that arrays C and D are provided as cell-interface data by the user. $MODE = 6$ is used like $MODE = 3$ with nonzero sources appearing only at the $NIND$ cells indicated in the first $NIND$ locations of the integer array $INDEX$. When $MODE = 1$, 3, 4, or 6, the array C is not used at all.

5-6. **C** and **D** are arrays of source datahaving different uses as described in item 4 above. The values of the source terms computed by SOURCES are passed to the subroutine

LCPFCT through the common block FCT_MISC in the real array SOURCE.

7-8. **D1** and **DN** are source data at the boundaries of the integration region.

Subroutine LCPFCT: –

The main subroutine LCPFCT integrates and updates the variables. It must be called for each of the conserved quantities to be integrated. Subroutine CNVFCT is treated exactly the same as LCPFCT with the one difference that the compression term in the continuity equation is left out. Thus CNVFCT really solves the 'advection' equation rather than the full continuity equation. The calling sequence for subroutine LCPFCT (CNVFCT) has nine arguments:

1. **RHOO** – A real array that stores the values of one of the conserved quantities ρ_i^o at the beginning of the timestep. The symbol ρ is used here to represent the mass density, momentum density, energy density, species density, or any other conserved quantity for which the fluid velocity v used in the previous call to VELOCITY is the appropriate convective velocity.

2. **RHON** – A real array that stores the values of ρ_i^n at the end of the timestep. RHOO and RHON may be the same array provided the values of RHOO do not need to be saved. (These arrays should be different if a two-step algorithm is being used.)

3–4. **I1** and **IN** are the first and last grid points integrated in the domain from cell $I1$ to IN.

The next four arguments, $SRHO1$, $VRHO1$, $SRHON$, $VRHON$, are real numbers used to define a general set of boundary conditions. LCPFCT uses guard cells on either end of the integrated domain to define boundary conditions. These guard cells are used to continue the calculation from the interior of the grid to the exterior of the grid and provide the missing data for the calculation on either end of the integrated domain. The guard-cell values are a linear combination of the value just inside the boundary and an externally imposed value.

5. **SRHO1** – The slope boundary condition factor for the guard-cell values adjacent to cell $I1$. Using $SRHO1$, the derivative of the solution can be specified on the first interface.

6. **VRHO1** – A constant to be added to the first guard cell values. The equation for the density value at guard cell $I1 - 1$ is $\rho_g = S_1 \times \rho_{I1} + V_1$.

7. **SRHON** – The slope boundary condition factor for the guard-cell values adjacent to cell IN. Using $SRHON$, the derivative of the solution can be specified on the last

Table 5.5 Arguments in Subroutines LCPFCT and CNVFCT

Call LCPFCT (RHOO, RHON, I1, IN, SRHO1, VRHO1, SRHON, VRHON, PBC)
Call CNVFCT (RHOO, RHON, I1, IN, SRHO1, VRHO1, SRHON, VRHON, PBC)

Variable Name	Type	Text Symbol	Meaning
RHOO	RA(N)*	ρ_i^o	Grid point densities at start
RHON	RA(N)	ρ_i^n	Grid point densities at end
I1	Integer	$I1or1$	Index of first cell of integration
IN	Integer	$INorN$	Index of last cell of integration
SRHO1	Real	S_1	First derivative boundary condition
VRHO1	Real	V_1	First constant boundary condition
SRHON	Real	S_N	Last derivative boundary condition
VRHON	Real	V_N	Last constant boundary condition
PBC	Logical		.true. = Periodic boundary conditions

* RA(N) stands for Real ARRAY of length N .

interface.

8. **VRHON** – A constant to be added to the last guard cell values. The equation for the density value at guard cell $IN + 1$ is $\rho_g = S_N \times \rho_{IN} + V_N$.

9. **PBC** – Declared a 'logical' variable with the value .true. for periodic boundary conditions and .false. otherwise. When periodic boundary conditions are used, the other four boundary condition variables (5–8 above) are ignored.

A more complete description of how to specify and use these variables along with examples of how to set up a variety of boundary conditions for gas dynamic problems is the subject of Section 6. Actual programs using some of these examples are given in the appendices.

The **Do** loops in LCPFCT can be identified with the corresponding equations in Section 3. Eq. (3.6) is combined with part of Eq. (3.9) in **Do 1**. **Do 2** combines the rest of Eq. (3.9) with eq. (3.10), the precomputed source terms, and with Eq. (3.11). **Do 3** corresponds to Eq. (3.14) and also computes the differences of the transported, diffused density later used in **Do 5**, the flux-correction formula Eq. (3.21). **Do 4** computes a number of the FCT terms also appearing in Eq. (3.21). **Do 5** calculates the new density profile $\{\rho_i^n\}$ of Eq. (3.22), returned as the output of LCPFCT.

In addition to the main sequence of subroutines, there are additional subroutines which give the LCPFCT library considerable added flexibility. These are: CNVFCT, a nonconservative form of FCT; ZEROFLUX and ZERODIFF, which turn off the advection and/or diffusion fluxes at specified interfaces; RESIDIFF, discussed above, to change the residual diffusion coefficient; and CONSERVE, which monitors the conservation of user-specified variables.

CNVFCT (RHOO, RHON, I1, IN, SRHO1, VRHO1, SRHON, VRHON, PBC) is a nonconservative convective form of the FCT solver which solves the advective equation

$$\frac{\partial \rho}{\partial t} + v\nabla\rho \;=\; \text{Sources} \;, \tag{5.1}$$

rather than the conservative form

$$\frac{\partial \rho}{\partial t} + \nabla\rho v \;=\; \text{Sources} \;. \tag{5.2}$$

CNVFCT serves roughly the same purpose as the LCPFCT subroutine and the argument list is identical to that of LCPFCT. CNVFCT finds its main usage when a mass fraction for a conserved density must be transported. Thus, if $f \equiv \rho_a/\rho_{total}$ is the ratio of a species density to the total density, f satisfies Eq. (5.1) rather than (5.2). Source terms are treated in the same way in both LCPFCT and CNVFCT and the sequence of auxiliary subroutine calls is also identical.

ZERODIFF (IND) is a specialized subroutine that allows the user to set to zero all diffusive and antidiffusive fluxes calculated in LCPFCT through an arbitrary number of specific cell interfaces. ZERODIFF does not alter the convective fluxes through the selected interfaces and is therefore useful in one-dimensional codes where the interfaces of a number of different materials are being tracked in a Lagrangian manner. ZERODIFF guarantees that there is no diffusion of the materials across the chosen interfaces. In multidimensional models, ZERODIFF may be used to ensure that nothing diffuses onto or off of the grid at a boundary where the incoming flux is known. Examples of the use of ZERODIFF are included in the test programs reprinted in the appendices. The call to ZERODIFF should be made just after the call to VELOCITY. A call to VELOCITY erases the effect of a call to ZERODIFF. The calling sequence for subroutine ZERODIFF has one argument:

1. **IND** – If positive, an integer index to the cell interface where the diffusive fluxes are to be set identically equal to zero. If negative, the user must set $NIND$ in common block FCT_NDEX to the number of interfaces to be zeroed and fill the first $NIND$ locations of the integer array $INDEX$ (also in FCT_NDEX) with the corresponding interface indices.

ZEROFLUX (IND) is similar to ZERODIFF, except that it sets to zero the convective fluxes as well as the diffusive and antidiffusive fluxes. The calling sequence is identical to that for ZERODIFF. ZEROFLUX is useful at solid walls. Examples of the use of ZEROFLUX are included in the test programs reprinted in the appendices. The call to ZEROFLUX should be made just after the call to VELOCITY. A call to VELOCITY erases the effect of a call to ZEROFLUX. The calling sequence for subroutine ZEROFLUX has one argument:

1. **IND** – If positive, an integer index to the cell interface where the diffusive fluxes are to be set identically equal to zero. If negative, the user must set $NIND$ in common block FCT_NDEX to the number of interfaces to be zeroed and fill the first $NIND$ locations of the integer array $INDEX$ (also in FCT_NDEX) with the corresponding interface indices.

CONSERVE (RHO, I1, IN, CSUM) is a useful utility subroutine that allows the user to monitor the conservation of variables. It calculates the conservation integral as a summation,

$$CSUM = \sum_{i=I1}^{IN} \Lambda_i^n \rho_i, \tag{5.3}$$

where the quantity summed can be any of the grid quantities $\{\rho_i\}$. CONSERVE computes the conservation integral using cell volumes from the last call to MAKEGRID. The calling sequence for subroutine CONSERVE has four arguments:

1. **RHO** – An array of real values of some grid quantity ρ_i^o for which the conservation integral is to be calculated.

2–3 **I1** and **IN** – The first and last cells to be included in the conservaton integral.

4. **CSUM** – A real variable where the summed quantity is to be stored.

LCPFCT is an exactly conservative algoritm in the sense that any physically conserved quantity summed over a fixed interval remains constant to within computer roundoff error. However, the user must beware of boundary conditions. Grid motion or fluxes out of the boundary result in a variable value of the conserved quantity determined by the flux through the two end boundary interfaces.

There are also three rather special purpose gridding routines: COPYGRID, which sets aside a copy of all the grid variables for later reuse without recomputation; SETGRID, which is used for polar coordinates; and NEWGRID, which reduces the number of geometrical calculations when the new grid interface locations change but the old ones remain the same. These subroutines are also reproduced on Appendix A and discussed briefly in Section 7 entitled "Additional Information".

5.3 Typical Calling Sequences

We now show how the various LCPFCT subroutines are interlinked to provide a complete calculation. Examples through simple Fortran test programs are also given in the appendices and summarized in the following tables. A typical sequence of calls for a one-dimensional gas dynamics problem on a fixed Eulerian grid is given in Table 5.6.

Table 5.6 Calls for One-Dimensional Eulerian Gas Dynamics Problems

† Calculate grid locations RADHN ($r^n_{i-\frac{1}{2}}$ for $i = 1, ..., N+1$); initialize variables
Call MAKEGRID (RADHN, RADHN, 1, N+1, 1)
† Begin timestep loop
† Compute half step interface velocity UH ($v^o_{i-\frac{1}{2}}$), cell PRES (P^o_i), and cell PV ($P^o_i v^o_i$)
Call VELOCITY (UH, 1, N+1, 0.5*DT)
Call ZERODIFF (1)
Call ZERODIFF (N+1)
Call LCPFCT (RHOO, RHON, 1, N, 1.0, 0.0, 1.0. 0.0, .false.)
Call SOURCES (1, N, 0.5*DT, 2, ONE, PRES, PRE1, PREN)
Call LCPFCT (RVXO, RVXN, 1, N, 1.0, 0.0, 1.0, 0.0, .false.)
Call SOURCES (1, N, 0.5*DT, 1, ZERO, PV, PV1, PVN)
Call LCPFCT (ERGO, ERGN, 1, N, 1.0, 0.0, 1.0, 0.0, .false.)
† Compute cell velocities, source terms, for the full step based on the half-step results (see GASDYN example in Appendix C)
Call VELOCITY (UH, 1, N+1, DT)
Call ZERODIFF (1)
Call ZERODIFF (N+1)
Call LCPFCT (RHOO, RHON, 1, N, 1.0, 0.0, 1.0. 0.0, .false)
Call SOURCES (1, N, DT, 2, ONE, PRES, PRE1, PREN)
Call LCPFCT (RVXO, RVXN, 1, N, 1.0, 0.0, 1.0, 0.0, .false)
Call SOURCES (1, N, DT, 1, ZERO, PV, PV1, PVN)
Call LCPFCT (ERGO, ERGN, 1, N, 1.0, 0.0, 1.0, 0.0, .false)
† Update velocities, and timestep and begin next timestep

† Indicates work the user must do.

Table 5.7 shows a sequence of calls appropriate to a Lagrangian fluid dynamics calculation (advancing the cell interface locations as well as ρ, ρU, and E). In these two

Table 5.7 Calls for One-Dimensional Lagrangian Gas-Dynamic Problem

† Begin timestep loop

† Calculate grid locations RADHO ($r^0_{i-\frac{1}{2}}$); initialize variables

† Compute new interface locations RADHN ($r^{\frac{1}{2}}_{i-\frac{1}{2}}$), interface velocities UH ($v^o_{i-\frac{1}{2}}$), and cell source terms PRES (P^o_i), and PV ($P^o_i v^o_i$) for the half step:

```
Call MAKEGRID ( RADHO, RADHN, 1, N+1, 1 )
Call VELOCITY ( UH, 1, N+1, 0.5*DT )
Call ZERODIFF ( 1 )
Call ZERODIFF ( N+1 )
Call LCPFCT (RHOO, RHON, 1, N, 1.0, 0.0, 1.0. 0.0, .false. )
Call SOURCES ( 1, N, 0.5*DT, 2, ONE, PRES, PLEFT, PRIGHT )
Call LCPFCT (RVXO, RVXN, 1, N, 1.0, 0.0, 1.0, 0.0, .false. )
Call SOURCES ( 1, N, 0.5*DT, 1, ZERO, PV, PVLEFT, PVRIGHT )
Call LCPFCT ( ERGO, ERGN, 1, N, 1.0, 0.0, 1.0, 0.0, .false. )
```

† Compute grid, velocities, and source terms for full step based on half-step results

```
Call MAKEGRID (RADHO, RADHN, 1, N+1, 1 )
Call VELOCITY ( UH, 1, N+1, DT )
Call ZERODIFF ( 1 )
Call ZERODIFF ( N+1 )
Call LCPFCT ( RHOO, RHON, 1, N, 1.0, 0.0, 1.0. 0.0, .false. )
Call SOURCES ( 1, N, DT, 2, ONE, PRES, PRE1, PREN )
Call LCPFCT ( RVXO, RVXN, 1, N, 1.0, 0.0, 1.0, 0.0, .false. )
Call SOURCES ( 1, N, DT, 1, ZERO, PV, PV1, PVN )
Call LCPFCT ( ERGO, ERGN, 1, N, 1.0, 0.0, 1.0, 0.0, .false. )
```

† Update new interface locations, velocities, and timestep and begin next timestep

† Indicates work the user must do.

examples cell-center pressure source terms are sent to Subroutine SOURCES for use with $MODE = 1$ and $MODE = 2$. In Subroutine GASDYN (Appendix C), interface pressure source terms are computed by the user, calling for $MODE = 3$ and $MODE = 4$ in Subroutine SOURCES. A multidimensional fluid dynamics program can be formed by time splitting the equations into successive calculations in each direction. The half and whole steps are performed in successive pairs with appropriate calls to MAKEGRID to change the grid for the coordinate directions. If the grid moves, MAKEGRID is called

Table 5.8 Calls for Two-Dimensional Gas-Dynamic Problem

† Begin new time step with the x coordinate direction integration ...
Call MAKEGRID (XCOORD, XCOORD, 1, NX+1, 1)
† Calculate velocities and source terms for the x-direction half step
Call VELOCITY (VX, 1, NX+1, 0.5*DT)
Call ZEROFLUX (1)
Call ZEROFLUX (NX+1)
Call LCPFCT (RHOO, RHON, 1, NX, 1.0, 0.0, 1.0, 0.0, .false.)
Call SOURCES (1, NX, 0.5*DT, 2, ONE, PRES, PRE1X, PRENX)
Call LCPFCT (RVXO, RVXN, 1, NX, 1.0, 0.0, -1.0, 0.0, .false.)
Call LCPFCT (RVY0, RVYN, 1, NX, 1.0, 0.0, 1.0, 0.0, .false.)
Call SOURCES (1, NX, 0.5*DT, 1, ZERO, PV, PV1X, PVNX)
Call LCPFCT (ERGO, ERGN, 1, NX, 1.0, 0.0, 1.0, 0.0, .false.)
† Calculate velocities and sources for whole step
† Repeat the last nine calls replacing 0.5*DT by DT
† Begin y-direction calculations
Call MAKEGRID (YCOORD, YCOORD, 1, NY+1, 1)
† Calculate y-direction velocities and sources for half step
Call VELOCITY (VY, 1, NY+1, 0.5*DT)
Call ZEROFLUX (1)
Call ZEROFLUX (NY+1)
Call LCPFCT (RHOO, RHON, 1, NY, 1.0, 0.0, 1.0, 0.0, .false.)
Call LCPFCT (RVXO, RVXN, 1, NY, 1.0, 0.0, 1.0, 0.0, .false.)
Call SOURCES (1, NY, 0.5*DT, 2, ONE, PRES, PRE1Y, PRENY)
Call LCPFCT (RVY0, RVYN, 1, NY, 1.0, 0.0, -1.0, 0.0, .false.)
Call SOURCES (1, NY, 0.5*DT, 1, ZERO, PV, PV1Y, PVNY)
Call LCPFCT (ERGO, ERGN, 1, NY, 1.0, 0.0, 1.0, 0.0, .false.)
† Repeat the above nine statements for the full timestep

† Indicates work the user must do.

with both the old and the new grid locations. A typical $x - y$ cartesian geometry model with hard wall boundaries is given schematically in Table 5.8. The daggers in the table indicate material the user must supply.

5.4 Summary of the Major LCPFCT Library Routines

LCPFCT has several subroutines that perform the four distinct tasks required to advance one or more general continuity equations a single timestep. Here we summarize these four subroutines and their associated arguments and calling sequences.

1. **MAKEGRID** (RADHO, RADHN, I1, INP, ALPHA) – sets up the finite volume computational grid at the start of the run or when changing cell definitions at the beginning of or during a timestep. Both the grid at the beginnning of the integration step and at the end of the integration step must be specified.

4. **VELOCITY** (UH, I1, INP, DT) – calculates all velocity dependent terms, the diffusion, and antidiffusion coefficients $\delta v_{i+\frac{1}{2}}, \nu_{i+\frac{1}{2}}, \mu_{i+\frac{1}{2}}$.

5. **SOURCES** (I1, IN, DT, MODE, C, D, D1, DN) – can be called once or repeatedly to build up composite sources terms of several types.

6. **LCPFCT** (RHOO, RHON, I1, IN, SRHO1, VRHO1, SRHON, VRHON, PBC) – takes the old densities RHOO on the old grid and calculates new densities RHON on the new grid using the flux-corrected transport (FCT) algorithm.

These four tasks described are programmed separately to eliminate unnecessary recalculation of quantities which appear in several different continuity equations having the same flow field (VELOCITY) or which must be calculated only when the grid is changed (MAKEGRID). The source terms (SOURCES) are treated separately because they are not used at all in many continuity equations. Source terms should always be computed immediately before they are used in a call to LCPFCT. LCPFCT then resets the source terms to zero just before returning to the users program each time it is called. For example, if the source terms for the energy equation are computed before the momentum equation is advanced, they will be lost after LCPFCT is called to advance the momentum equation. In addition the momentum equation would then be solved incorrectly because it includes the energy source terms.

6. BOUNDARY CONDITIONS

Boundary conditions, distinct from the LCPFCT solution algorithm, are needed to model both confined and unconfined computational domains. When the system modeled is effectively unconfined, an infinite volume must be represented with only a finite number of degrees of freedom. When the system is confined, the effects of realistic walls and flexible interfaces with regions of inflow and outflow must be treated. Boundary conditions for computational fluid dynamics have been discussed extensively – see, for example, Grosch and Orszag (1977), Turkel (1980), and Kutler (1982). Oran and Boris (1987) discuss some of the general problems of applying boundary conditions to finite-difference and finite-volume models. Specifying physically accurate outflow conditions for multidimensional regions of subsonic flow receives the most attention. Recently Thompson (1987,1990), Givoli (1991), and Poinsot and Lele (1992) have discussed methods for this problem. Grinstein (1993) has considered these open outflow boundary conditions specifically for the FCT algorithms used here.

Depending on the physical modes in the system being simulated, accurate treatment of the boundary conditions will differ greatly. In hyperbolic systems that simulate convection and acoustic phenomena, waves may travel much faster than the convective flows in which they propagate. These waves carry information through the medium in all directions and thus information can enter the computational grid as well as leave it. In parabolic systems, the information flow through the computational domain is generally one sided so the solution can be advanced preferentially in one direction. Some supersonic, compressible flows have all characteristics moving in one direction so that one-sided or direction-biased equations can be used to find suitable boundary conditions.

There are three distinct ways to implement boundary conditions in numerical models (Oran and Boris, 1987):

1. Expand the continuum fluid variables in a linear superposition of expansion functions with the boundary conditions built into each of them, so that any combination automatically satisfies the boundary conditions. Although expansions are used in many methods, this approach cannot be applied systematically to the FCT algorithms.

2. Develop separate finite-difference formulas for boundary cell values which reflect the formulas used in the interior of the mesh combined with auxiliary relations to determine the values of the grid variables which would lie outside the computational domain. These formulas often involve simple analytic formulations for the boundary variables that use information about the behavior of the system near and at the wall or as it approaches infinity.

3. Develop extrapolations from the interior to *guard* or *ghost* cells outside the computa-

tional domain that continue the mesh a distance beyond the domain boundary. These *ghost* boundary cells allow cells on the domain boundary to be treated as interior cells, often with appreciable simplification in the programming.

Of these methods, the third, defining guard-cell values calculated separately, is the easiest to use and is therefore adopted in LCPFCT. Figure 6.1 shows a two-dimensional uniform grid whose boundary has been outlined by thicker lines. The grid has N_x cells in the x-direction, extending from x_L to x_R, and N_y cells in the y-direction, extending from y_B to y_T. Two rows of cells in dashed lines are shown surrounding the computational grid, the so-called *guard cells* or *ghost cells*. Most applications of the LCPFCT subroutines require the user to provide conditions for one layer of guard cells; any second or third layers used implicitly in higher differences are calculated inside the LCPFCT routines from the user-provided formulas.

Using special finite-difference formulas near boundaries is usually equivalent to defining variable values in guard cells. In LCPFCT, the user provides the coefficients of formulas that define the guard cell values from the current boundary cell quantities. Rather than specifying fixed values at the beginning of the timestep, this approach allows the guard cell values to be updated repeatedly as the fluid quantities in the boundary cell change during the timestep.

By assigning appropriate values to the guard cells, the fluid elements in the domain interior can be made to feel appropriate idealized external or reflecting wall conditions as if there were no boundary at all. When the values of variables assigned to guard cells are stored in the same arrays as the interior variables, the calculation can be advanced on the entire grid using a single set of vectorizable finite-difference equations.

To reduce the amount of extra memory required in multidimensional simulations, guard cells are defined by LCPFCT for only one row at a time. For example, a $20 \times 20 \times 20$ grid uses 8000 memory locations per variable. If this grid is extended two guard cells in all directions, the resulting $24 \times 24 \times 24$ grid has 13,824 cells, almost a factor of two increase with no corresponding improvement in resolution! Because LCPFCT redefines the guard-cell values for each row being integrated, maximum use of the available computer memory can be made for spatial resolution.

6.1 Representation of Boundary Conditions in LCPFCT

The specific formulation of the boundary conditions used in LCPFCT includes symmetry, antisymmetry, periodicity, and inflow-outflow and is given by

$$\rho_G^n \;=\; S_{EBC}\,\rho_{IE}^n + V_{EBC}, \tag{6.1a}$$

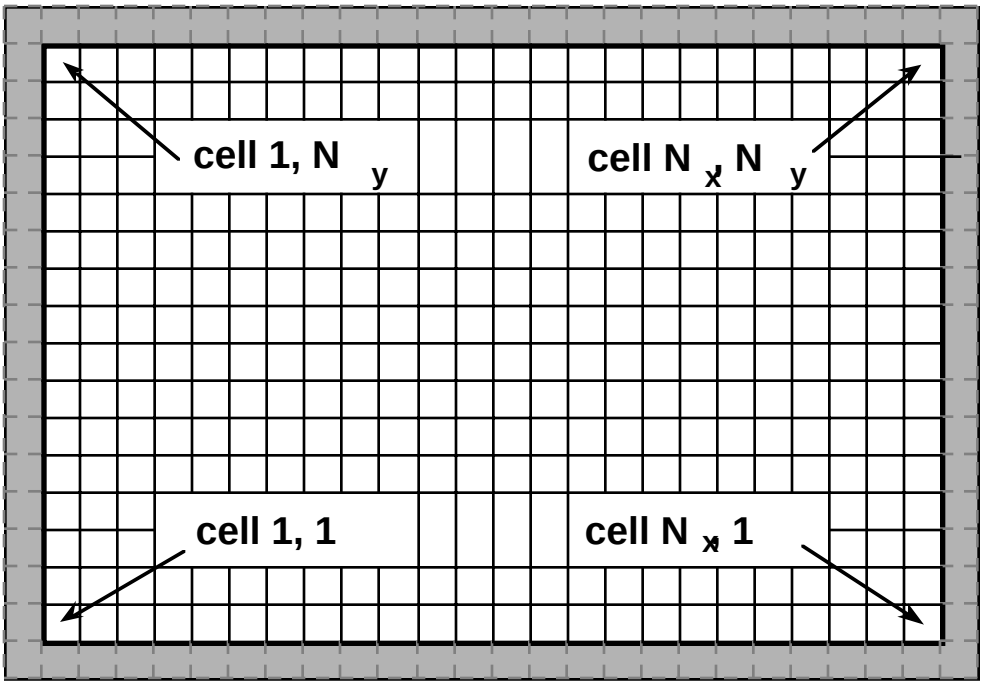

Figure 6.1 A two-dimensional computational domain with one row of (shaded) guard cells. Values of variables must be specifed in these cells to model various boundary conditions on the physical system in order that the variable values can be updated on the interior cells.

or, for periodic boundary conditions,

$$\rho_G^n \;=\; \rho_{NO}^n \;, \tag{6.1b}$$

where ρ^n indicates the value of one of the conserved densities $\rho(x)$ at timestep n. Thus the variable ρ stands for mass density, momentum density, energy density or a chemical species density. The subscript IE stands for either $I1$ or IN depending on which End of the system is being described, i.e., the boundary at the 1st or the Nth cell. The subscript G indicates the guard cell for the computational domain, either $I1-1$ when IE is replaced by $I1$, or $IN+1$ when IE is replaced by IN.

The quantity S_{EBC} is the Slope boundary condition factor for Either Boundary Condition, the first cell boundary condition (when EBC is replaced by $BC1$), or the last cell boundary condition (when EBC is replaced by BCN). This factor multiplies the current value just inside the boundary of the computational domain, ρ_{IE}, to get the linearly varying component of the corresponding guard cell value. The quantity V_{EBC} is specified by the user external to the LCPFCT subroutine as a Value added to the guard-cell value. In a physical situation where the guard cell value does not change regardless of

the adjacent value ρ_{IE}, the slope factor S_{EBC} would be set to zero and V_{EBC} in Eq. (6.1) is given the correct value. V_{EBC} might be the fixed temperature of a hot wall, for example, or the value of the mass density flowing into the computational domain at supersonic speed.

Each of the ideal confined and symmetry boundary conditions treated below and described in Table 6.1 and Table 6.2 can be obtained by choosing appropriate values of $+1$ or -1 for S_{EBC} and setting the added constant V_{EBC} to zero. All of the continuitive inflow-outflow boundary condition cases treated in Section 6.3 in conjunction with Table 6.3, Table 6.4, and Table 6.5 can also be obtained by manipulating the values of S_{EBC} and V_{EBC} sent to the LCPFCT program. Equation (6.1) also allows periodic boundary conditions to be implemented by choosing P_{BC} = .true. rather than .false. and by setting both S_{EBC} and V_{EBC} to zero. In this case, ρ^n_{NO} is the value of the convected conserved variable at the Other boundary. $NO = I1$ when IE is replaced by IN and $NO = IN$ when IE is replaced by $I1$.

6.2 Boundary Conditions for Confined Domains

Ideal Symmetry or Nonboundary Conditions

The easiest way to treat boundary conditions accurately and consistently is to eliminate them. This can be done in regions where a symmetry condition exists. The guard-cell values are predicted exactly from values at corresponding (symmetric) interior locations. Symmetry conditions can often be applied in the interior of a system as well as at a boundary. A system may have a natural symmetry plane or line, such as the axis of an axisymmetric flow. Often a symmetry line is a good approximation to the three-dimensional system, as in the case of two equal and opposite impinging jets.

Figure 6.2 shows the right boundary and the last few cells of a one-dimensional grid terminating at a cell interface, as adopted in LCPFCT. This is called a *cell-interface boundary*. The edge of the grid in the figure is shaded to indicate a wall and a mirror (image) system is shown in the several guard cells beyond the boundary. Similar figures could also be drawn for the left boundary and the first few cells of the grid.

When the given grid locations specify the cell interfaces, the piecewise constant interpretation of the physical variable throughout the domain is natural even though there are apparent discontinuities at the cell interfaces. In this representation, cells adjacent to the boundary are complete. The boundary fluxes enter and exit the system at interfaces where the geometric areas as well as the variable values must be known. Conservation is ensured by controlling the fluxes at these cell interfaces.

In a piecewise linear representation, the cell-interface locations are generally interpolated from the cell-center locations. The piecewise linear format looks smoother, a major

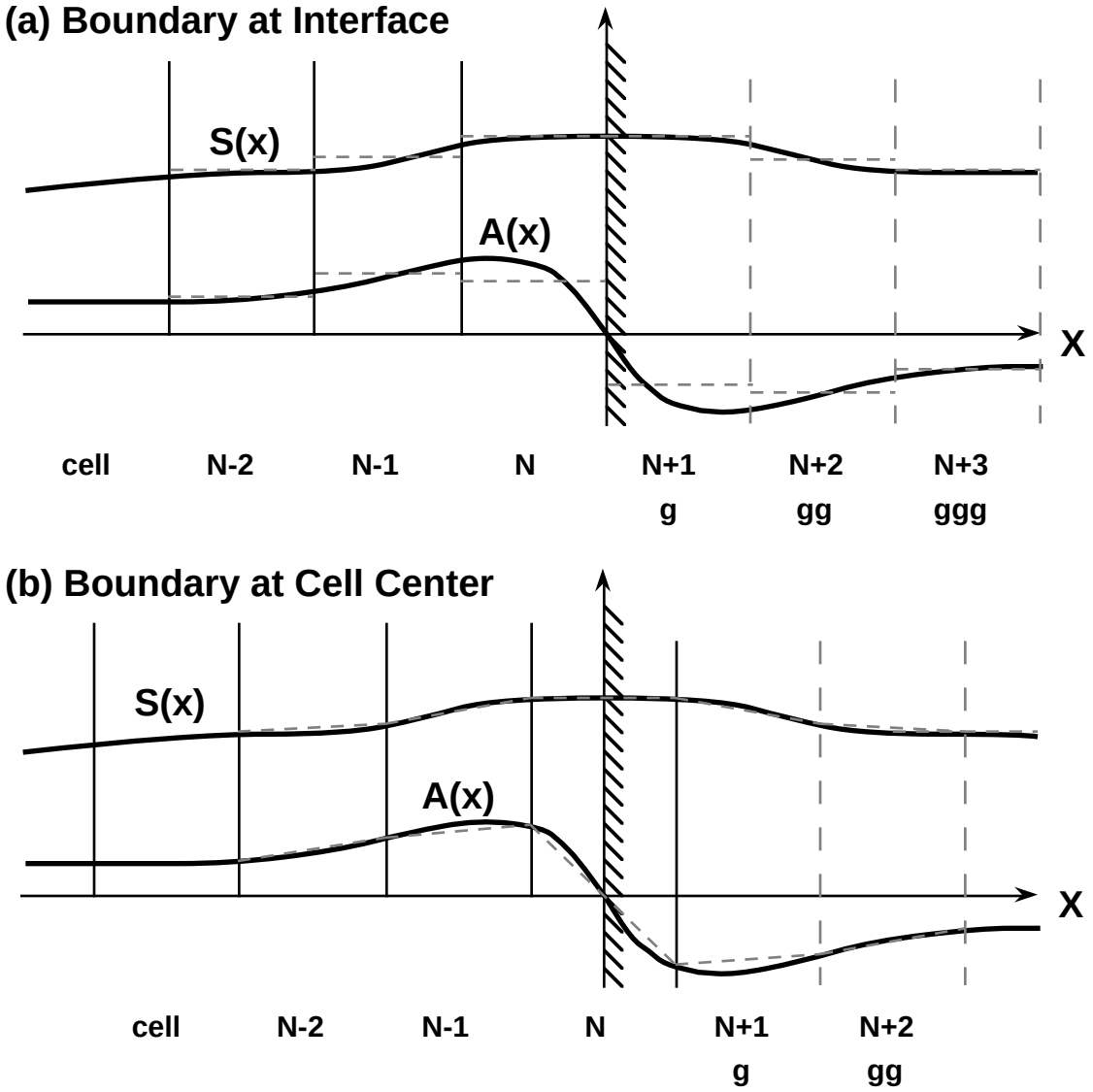

Figure 6.2 Different applications of symmetry boundary conditions with guard cells. The interior of the computational domain is on the left of the shaded band, and the guard cells are on the right. $S(x)$ is a symmetric function and $A(x)$ is an antisymmetric function. (a) Piecewise constant representation with a cell interface boundary (used by LCPFCT). (b) Piecewise linear representation with a cell-centered boundary (not used by LCPFCT).

advantage when the time comes to show results. Most contour-plotting routines interpret the function values as specified at the grid of points with a linear approximation implied in the cells between. The resulting contours are visually quite smooth, showing discontinuities in direction but not in value at multidimensional cell interfaces. In fact, piecewise constant representations are often presented graphically as if they were piecewise linear.

Table 6.1. Guard-Cell Formulas: Symmetric (S), Antisymmetric (A), Periodic (P)

Boundary at Interface I1*	Boundary at Interface IN+1*
$S_{I1-1} = S_{I1}$	$S_{IN+1} = S_{IN}$
$S_{I1-2} = S_{I1+1}$	$S_{IN+2} = S_{IN-1}$
$A_{I1-1} = -A_{I1}$	$A_{IN+1} = -A_{IN}$
$A_{I1-2} = -A_{I1+1}$	$A_{IN+2} = -A_{IN-1}$
$P_{I1-1} = P_{IN}$	$P_{IN+1} = P_{I1}$
$P_{I1-2} = P_{IN-1}$	$P_{IN+2} = P_{I1+1}$

$*$ Throughout LCPFCT boundaries are defined at cell interfaces.

The piecewise constant format is ideally suited for presentation as pixel plots where a rectangle of color is filled in for each cell of a variable.

In LCPFCT, the position of the cell interfaces are specified rather than cell-centers. This change from the previous version, ETBFCT, allows us to capture the added simplicity and flexibility of controlling the boundary fluxes exactly. Figure 6.2 shows two fluid variables, $S(x)$ and $A(x)$. $S(x)$ is symmetric and $A(x)$ is antisymmetric with respect to the bounding cell interface. In addition to $A(x)$ and $S(x)$, we also consider a periodic function $P(x)$. Periodic boundary conditions are relatively easy to treat numerically as they are simple variants of the symmetry conditions in which interior values somewhere are used to set guard-cell values elsewhere. The values of the variables at cell $N + 1$ are equal to those at cell 1 in a periodic system. Periodic boundary conditions arise in circular systems, such as stacks of turbine blades, cylindrical systems, spherical systems, or in idealized approximations to small segments of large Cartesian systems. We generally assume that the guard cells are the same size as the corresponding cells just inside the boundary. Table 6.1 lists simple guard-cell formulas for symmetric, antisymmetric, and periodic variables for the interface-centered boundaries used in the LCPFCT modules. The formulas include a second layer of guard cells for completeness though LCPFCT uses only the first row as shown in Figure 6.1.

When flow with both normal and tangential components is directed against an ideal wall, the physical conditions in the guard cells can be found from the values in the nearby interior cells. This is important for bounded Euler flows and for high Reynolds-number viscous flows, in which the boundary layers are approximated by additional phenomenologies. Both symmetry and antisymmetry conditions must be applied at such a wall. The

lowest-order condition is symmetric, as the density, temperature, and pressure have zero slope at the wall. The tangential velocity for *free slip* conditions is symmetric but the normal velocity is antisymmetric. The guard-cell values of variables for flow against a wall are given in Table 6.2.

Table 6.2. Free-Slip Flow Confined by an Insulating Hard Wall

Free Slip Hard Wall*	**S**lope S_{EBC}	**V**alue V_{EBC}
Density, Temperature, and Pressure:		
$\rho_G = \rho_{IE}$ $T_G = T_{IE}$ $P_G = P_{IE}$	$S_\rho = 1$	$V_\rho = 0$
Momentum Parallel to Integration Direction (Perpendicular to Wall):		
$\rho_G v_{\parallel G} = -\rho_{IE} v_{\parallel IE}$	$S_\parallel = -1$	$V_\parallel = 0$
Momentum Transverse to Integration Direction (Along Wall):		
$\rho_G v_{\perp G} = \rho_{IE} v_{\perp IE}$	$S_\perp = 1$	$V_\perp = 0$
Total Fluid Energy Density:		
$E_G = E_{IE}$	$S_E = 1$	$V_E = 0$
Species Number Densities:		
$n_{i,G} = n_{i,IE}$	$S_i = 1$	$V_i = 0$

* Subscript G refers to the guard cell, either cell $I1 - 1$ or cell $IN + 1$. Subscript IE refers to the boundary or end cell of the computational domain, *E*ither cell $I1$ or cell IN, for the particular side of the grid being considered.

Sometimes values of the physical variables on the boundary cannot be determined directly by applying finite-difference equations with symmetry conditions. It is often necessary to develop modified boundary formulas that depend on the information from the interior. This allows representation of more physically complex situations such as viscous and turbulent boundary layers. The extrapolation formulas generally use one-sided relations where interior simulation data are combined with phenomenological exterior or boundary layer data.

Boundary Layers and Surface Phenomenologies

Boundaries often model interfaces between two different phases or two materials. A solid container with a gas inside may be chemically inert, thermally insulating, absolutely rigid, and perfectly smooth. If these approximations are acceptable, simplified symmetry and guard-cell algorithms can be used, as indicated above. Many phenomena, however, depend on the nonideal nature of the boundaries.

Analyses of numerical boundary layers and other surface models fill books and include subgrid phenomenologies for representing other types of physics not generally resolved in the simulation. For example, catalytic reactions at walls and thermal boundary layers can be modeled by surface phenomenologies. As another example, when heat transfer to or from a wall is high enough, condensation, evaporation, or even ablation can occur.

Developing such surface phenomenologies requires satisfying the conservation equations at and through the boundaries. This means that fluxes of mass, momentum, energy, and enthalpy, which enter or leave the computational domain through the boundaries, must exactly equal the fluxes to the exterior world. Causality and the conservation laws provide constraints in these problems. For example, if a thermal boundary layer forms in a gas next to a cold metal wall, the thermal energy moving from the last cell into the wall should not exceed the thermal capacity of that cell. If the temperature scale lengths in the gas are resolved and can be estimated in the metal wall, simple energy interchange approximations between the interior cells and the exterior are adequate to define suitable interface fluxes in the LCPFCT modules.

If catalytic surface reactions are occurring, the corresponding boundary condition must estimate the amount of the reactant that encounters the boundary and evaluate the probabilities of each possible reaction pathway. When the grid is finely resolved at the wall so that molecular diffusion scale lengths can be resolved, treating the wall interaction is relatively straightforward. When the cells are too large, however, the reacting species seem to be spread throughout the last two or three cells by LCPFCT even though they should actually be concentrated near the wall. Thus purely surface effects will appear as volume averages, leading to spurious numerical rates.

A similar effect arises from viscosity. The usual macroscopic treatment of the Navier-Stokes equation assumes the *zero-slip condition*, which means that the tangential velocity at the wall is zero. On a microscopic scale, molecules rebounding from a rigid wall are assumed statistically to lose memory of their tangential direction prior to collision with the wall. Thus they have equal probability of scattering forward or backward relative to the flow of the fluid far from the wall. A thin laminar boundary layer forms near the wall. From the macroscopic fluid point of view, this boundary layer develops on a very small

Figure 6.3 Schematic of a grid where the physical boundary layer is substantially smaller than the computational cell adjacent to the wall.

spatial scale which is usually very expensive to resolve. Again, a subgrid phenomenology is often used to satisfy the physical conditions at this rigid surface. When the computational cells are large, the tangential fluid momentum deficit due to this boundary layer is small even in the cells adjacent to the wall. Not much fluid is slowed down. However, some fluid is moving at almost zero velocity, as shown in Figure 6.3.

The normal component of velocity must vanish at the wall on both the macroscopic and the microscopic scale, a condition satisfied by an ideal symmetry condition in LCPFCT. However, in a viscous fluid, the tangential flow at the surface must also be zero. This additional tangential zero-slip condition is a complicating factor requiring explicit representation. It also requires resolution of the viscosity and viscous scales or a good "law of the wall" phenomenology. The velocity at the wall is rigorously zero, and viscosity diffuses the momentum deficit from the boundary layer into the free flow further from the wall. To actually resolve this boundary layer, very small cells are needed normal the wall. These small cells, in turn, impose a severe timestep restriction unless the equations are integrated implicitly. In an LCPFCT calculation based on a primitive variable formulation, the drag at the wall can be modeled by subtracting momentum from the fluid near the boundary, extracting just the right amount of parallel momentum from the two or three layers of cells adjacent to the wall every timestep so that the velocity at the wall approaches zero. This mometum deficit can in turn be obtained from some appropriate boundary layer model.

6.3 Continuitive Boundary Conditions for Unconfined Domains

Simulating an unconfined flow requires representing an effectively infinite region as a finite computational domain. The boundary conditions must transmit information to and from the entire outside world, properly absorbing any signals coming from the computational domain. Systems coupled to an exterior domain can be rigorously computed on a bounded domain only when the variables and coefficients in the problem become constant at infinity. When these coefficients are not constant, approximations are always needed, and there will be an inaccuracies having nothing to do with the accuracy of the interior methods.

One approach is to map the infinite region into a finite domain by analytically redefining the independent spatial variables. The problem with this approach is that finite-wavelength components are not resolved properly near the edges of the transformed grid. The spatial resolution of the grid becomes inadequate to propagate information at wavelengths of interest (see, for example, Grosch and Orszag, 1977). Another approach is to truncate the simulated domain at a finite distance and analytically model the influence of the exterior world on the domain boundaries. The shorter wavelengths can now propagate up to the boundaries, but they are partially reflected in a nonphysical way if an exact analytic condition is not available.

We recommend a combination of these approaches. First, the cells should be made progressively larger away from the central region of interest, thereby pushing the computational problems far away. By stretching the cells near the edges, or equivalently, by making them small only in regions of interest, the computational domain can be quite large without a corresponding increase in computer storage. Errors still arise from lack of knowledge of the solution in the exterior region. However, these affect the solution only weakly, and only after a delay for the numerical boundary condition influences to reach the central region. LCPFCT was specifically formulated to allow variable cell sizes, but the order of accuracy decreases in regions where the cells are changing rapidly. Cell stretching should be limited to 10–20% per cell in any direction to control these inaccuracies. Multidimensional cells with large aspect ratios should also be avoided whenever possible.

In a problem with an open boundary, analytic or phenomenological models can be used to approximate the values of the simulation variables in the region outside of the computed domain. The coefficients appearing in Eq. (6.1) are then defined using these models to convey the desired approximate guard cell dependences to the numerical integration modules. This is done so the most recently computed values near the edge of the domain can be combined with the auxiliary information about the exterior behavior of the solution at each stage of the integration. In this way, potentially unstable numerical extrapolation off the edge of the computational domain is effectively replaced by a more stable interpolation.

Subsonic and Supersonic Inflow Boundary Conditions

Inflow boundary conditions are often as difficult to implement properly as outflow boundary conditions, even though it would seem that everything is in fact known about the fluid entering the system. These difficulties arise because there are characteristics which can move outward at an inflow boundary which is subsonic. Thus we may not actually be in a position to specify everything about the fluid entering the system. These uncertainties and errors are particularly important because the fluid entering the system with errors arising from nonphysical boundary conditions stays resident for a long time. Thus the errors made on inflow boundaries often build up more rapidly and pollute the solution more completely than outflow boundary condition errors.

Table 6.3 summarizes useful choices of S_{EBC} and V_{EBC} appearing in Eq. (3.7) that can be used to implement subsonic compressible inflow in LCPFCT. Many variations of these choices are possible and may actually work better than these suggestions for particular flows or parameter regimes. Nevertheless, because we have been able to perform a wide range of problems more than adequately with these choice, we recommend at least trying them in your application.

The physical reasoning behind the definitions made in Table 6.3 is to specify two of the three inflowing quantities (in one dimension) corresponding to the two fluid characteristics entering the system. The two quantities chosen are the incoming mass flux and fluid entropy. This allows the propagation of isentropic pressure pulses (weak acoustic waves) upstream into the oncoming flow. The pressure of the incoming fluid varies so there is no pressure gradient at the bounding cell interface. This requires a corresponding change in the inflow density. This density variation in the inflow in turn requires a velocity change to keep the mass flux constant. If a pressure pulse from downstream causes the boundary pressure to go up, the incoming fluid compresses and slows down. The incoming energy density is computed from the calculated values of the incoming density, velocity, and pressure (Table 6.3).

This combination of effects appears to mimic a realistic inflow plenum quite well, preventing numerical effects from driving large, unphysical waves into the system after a long time.

Supersonic inflow boundary conditions are somewhat simpler since all of the fluid characteristics are entering the system from the guard cell and can therefore be assumed to be known. The velocity and pressure at the supersonic inflow boundary are also known so all of the input to the VELOCITY and SOURCES subroutines are known and the slope boundary condition factors in the calls to LCPFCT are set to zero.

Table 6.3. Inflow Boundary Condition Parameters for LCPFCT (P_{BC} =.false.)

Subsonic Inflow Conditions*	Slope S_{EBC}	Added Value V_{EBC}
Pressure and Mass Density:		
$P_G = P_{IE}$		
$\rho_G = V_\rho$	$S_\rho = 0$	$V_\rho = \rho_{inflow}\left(\frac{P_{IE}}{P_{inflow}}\right)^{\frac{1}{\gamma}}$
Momentum Density Parallel to Integration (normal to the boundary):		
$\rho_G v_{\parallel G} = \rho_{inflow} v_{\parallel inflow}$	$S_\parallel = 0$	$V_\parallel = \rho_{inflow} v_{\parallel inflow}$
Momentum Density Tangential to Boundary (along the boundary):		
$\rho_G v_{\perp G} = \rho_{inflow} v_{\perp inflow}$	$S_\perp = 0$	$V_\perp = \rho_{inflow} v_{\perp inflow}$
Total Fluid Energy Density:		
$E_G = S_E\, E_{IE} + V_E$	$S_E = 0$	$V_E = \frac{P_G}{(\gamma-1)} + \frac{1}{2}\rho_G({v_{\parallel G}}^2 + {v_{\perp G}}^2)$

Supersonic Inflow Conditions*	Slope S_{EBC}	Added Value V_{EBC}
Mass Density, Total Fluid Energy Density:		
$\rho_G = \rho_{inflow}$	$S_\rho = 0$	$V_\rho = \rho_{inflow}$
$E_G = E_{inflow}$	$S_E = 0$	$V_E = E_{inflow}$
Momentum Density Parallel to Integration (normal to the boundary):		
$\rho_G v_{\parallel G} = \rho_{inflow} v_{\parallel inflow}$	$S_\parallel = 0$	$V_\parallel = \rho_{inflow} v_{\parallel inflow}$
Momentum Density Tangential to Boundary (along the boundary):		
$\rho_G v_{\perp G} = \rho_{inflow} v_{\perp inflow}$	$S_\perp = 0$	$V_\perp = \rho_{inflow} v_{\perp inflow}$

∗ Subscript G refers to the guard cell, either cell $I1-1$ or $IN+1$. Subscript IE refers to the boundary or end cell of the computational domain, either $I1$ or IN, for the particular side of the grid being considered. The quantities subscripted "inflow" are specified by the user external to the LCPFCT modules.

Subsonic, Choked, and Supersonic Outflow Boundary Conditions

The difficulty with outflow boundary conditions is nonphysical reflection of characteristics which should leave through the boundary. In all but the simplest linear problems, the boundary algorithms can only be approximate because inward and outward propagating waves and pulses become locally indistinguishable in the nonlinear fluid dynamic equations.

Monotone convection algorithms such as LCPFCT can extrapolate the flow parameters off the edge of the system by setting guard-cell values as described in section (6.1). In two- or three-dimensional flows, this extrapolation should be done along the flow lines as a better approximation to following the characteristics though it is usually done normal to the boundary to comply with the data structure and logic of the split-step approximation to multidimensions. In this latter case it makes sense to differentiate regions where the outflow is predominantly normal to the boundary from those where it is nearly tangential.

Extrapolation is often unstable in linear convection algorithms for which positivity is not guaranteed. One of the major advantages of monotone methods is their ability to operate stably and reasonably accurately using such simplifications of characteristic analysis. By breaking the fluid disturbance into its constituent characteristics, however, and extrapolating each of these out to the guard cells, more stable, accurate outflow conditions are obtained even for monotone convection algorithms.

Table 6.4 describes simple boundary conditions for a fluid flowing off the edge of a finite computational domain. Used with a monotone method, these formulas for guard cells take into account the continuity of flow in the vicinity of the boundary. The quantity τ is a characteristic time for relaxation of the pressure to its ambient value at infinity, for example a characteristic system size divided by the average sound speed. The lowest-order extrapolation for guard-cell values uses the adjacent cell values for the corresponding guard cells. That is, $\tau = \infty$. The next higher-order extrapolation uses the two cells just inside the boundary to extrapolate linearly to the guard cells. The values of the variables at or near infinity, such as ρ_∞, feed information about the external world into the computational domain. Without these terms, that is, with $\tau = \infty$, a simulation cannot relax to the asymptotic pressure, and this leads to growing errors.

Conditions for Flow Roughly Parallel to an Open Boundary

There is a final case to be considered which occurs when the boundary of the computational domain is physically unconfined and the flow at infinity is roughly parallel to the boundary. This is neither the inflow nor the outflow situation as described in the two subsections because the actual fluid velocity perpendicular to the boundary is neither predominantly inward nor predominantly outward. Depending on fluctuations and sound waves near the boundary the fluid could either "breathe" in or out so the relative value of the tangential velocity is zero and yet the expected momentum and kinetic energy of the fluid are quite large. In this case the LCPFCT routines should be employed with the boundary condition parameters as defined in Table 6.5.

Table 6.4. Outflow Boundary Condition Parameters for LCPFCT (P_{BC} =.false.)

Subsonic & Supersonic Outflow*	S_{EBC}	V_{EBC}
Outflow Interface Pressure and Velocity:		
$v_{\parallel interface} = v_{\parallel IE}[1 - \frac{\Delta t}{\tau}]$	$P_{interface} = P_{IE}[1 - \frac{\Delta t}{\tau}] + \frac{\Delta t}{\tau} P_\infty$	
Mass Density:		
$\rho_G = S_\rho \, \rho_{IE} + V_\rho$	$S_\rho = [1 - \frac{\Delta t}{\tau}]$	$V_\rho = \frac{\Delta t}{\tau} \rho_\infty$
where τ is the problem-dependent time constant for relaxation of ρ to ρ_∞		
Momentum Density Parallel to Integration (normal to the boundary):		
$\rho_G v_{\parallel G} = S_\parallel \rho_{IE} v_{\parallel IE} + V_\parallel$	$S_\parallel = [1 - \frac{\Delta t}{\tau}]$	$V_\parallel = 0$
Momentum Density Tangential to Boundary (along the boundary):		
$\rho_G v_{\perp G} = S_\perp \rho_{IE} v_{\perp IE} + V_\perp$	$S_\perp = [1 - \frac{\Delta t}{\tau}]$	$V_\perp = 0$
Total Fluid Energy Density:		
$E_G = S_E \, E_{IE} + V_E$	$S_E = [1 - \frac{\Delta t}{\tau}]$	$V_E = \frac{\Delta t}{\tau} E_\infty$
Different values of τ may be appropriate for density, momentum, energy, etc.		

Choked Outflow Conditions*	S_{EBC}	V_{EBC}
Choked Interface Pressure:		
$P_{interface} = P_{IE}$		
Choked Outflow Interface Tangential Velocity:		
$v_{\parallel interface} = v_{s\,interface} = \left(\frac{\gamma P_{IE}}{\rho_{IE}}\right)^{1/2}$		
Mass Density, Total Fluid Energy Density:		
$\rho_G = S_\rho \, \rho_{IE} + V_\rho$	$S_\rho = [1 - \frac{\Delta t}{\tau}]$	$V_\rho = \frac{\Delta t}{\tau} \rho_\infty$
$E_G = S_E \, E_{IE} + V_E$	$S_E = [1 - \frac{\Delta t}{\tau}]$	$V_E = \frac{\Delta t}{\tau} E_\infty$
Tangential and Parallel Momentum Densities (as above):		

$*$ Subscript G refers to the guard cell, either cell $I1 - 1$ or $IN + 1$. Subscript IE refers to the boundary or end cell of the computational domain, either $I1$ or IN, for the particular side of the grid being considered. The quantities subscripted "∞" are specified by the user external to the LCPFCT modules and represent the ambient fluid variables far from the boundary in question.

Table 6.5. Parallel Flow Open Boundary Parameters for LCPFCT ($P_{BC} = 0$)

Parallel Open Boundary Flow*	S_{EBC}	V_{EBC}
Tangential Velocity and Pressure at Interface:		
$v_{\parallel interface} = v_{\parallel IE}$	$P_{interface} = P_{IE}$	
Mass Density, Species Number Densities:		
$\rho_G = S_\rho \, \rho_{IE} + V_\rho$	$S_\rho = [1 - \frac{\Delta t}{\tau}]$	$V_\rho = \frac{\Delta t}{\tau} \rho_{external}$
where τ is the relaxation time constant of ρ to $\rho_{external}$		
Momentum Density Parallel to Integration (normal to the boundary):		
$\rho_G v_{\parallel G} = \rho_{IE} v_{\parallel IE}$	$S_\parallel = [1 - \frac{\Delta t}{\tau}]$	$V_\parallel = \frac{\Delta t}{\tau} \rho_{external} v_{\parallel external}$
Momentum Density Tangential to Boundary (along the boundary):		
$\rho v_{\perp G} = \rho_{IE} v_{\perp IE}$	$S_\perp = 1$	$V_\perp = 0$
Total Fluid Energy Density:		
$E_G = S_E \, E_{IE} + V_E$	$S_E = [1 - \frac{\Delta t}{\tau}]$	$V_E = \frac{\Delta t}{\tau} E_{external}$
Different values of τ may be appropriate for density, momentum, energy, etc.		

$*$ Subscript G refers to the guard cell, either cell $I1 - 1$ or $IN + 1$. Subscript IE refers to the boundary or end cell of the computational domain, either $I1$ or IN, for the particular side of the grid being considered. The quantities subscripted "external" are specified by the user before calling the LCPFCT module and represent the value of the fluid variables external to the boundary in question.

7. ADDITIONAL INFORMATION

This section discusses some additional information about the use of the LCPFCT package which may be of use in some situations and which should help to extend a user's understanding of the algorithms and the implementation presented here. Topics considered are additional routines, more about boundary conditions through the use of the subroutines ZEROFLUX and ZERODIFF, flexibility in choosing interface averages, and a brief discussion of things an advanced user might want to try with the flux-correction formula.

There are a several additional subroutines in the LCPFCT package which provide flexibility but do not necessarily have to be fit into the normal sequence of FCT calls.

RESIDIFF (DIFFA) allows the user to add a small amount of residual diffusion to the solution by making the effective antidiffusion coefficients $\{\mu_{i-\frac{1}{2}}\}$ ($MULH$ in

FCT_VELO) be slightly smaller that the diffusion coefficients $\{\nu_{i-\frac{1}{2}}\}$ ($NULH$). To do this in LCPFCT, the magnitude of the sign function array, called $FSGN$ in **Do 2**, need not be exactly unity. A smaller value, for example the default value 0.999, is transferred to the antidiffusion coefficients and allows a slight residual diffusion that helps maintain smoother solutions for systems of gas dynamic equations. $DIFF1$ is carried in common block FCT_MISC and can be changed by a call to subroutine RESIDIFF, whose only argument is the new value of $DIFF1$. The value of $DIFF1$ stays at the newly set value until a new call to RESIDIFF is made. This facility is employed in a couple of the test problems. The value of $DIFF1$ is defaulted to 0.999 in a block data initializing routine and values less than 0.99 are not recommended. In some applications where very small Courant numbers are expected, this may lead to appreciable numerical diffusion in some regions and the user may want to be set $DIFF1$ to unity by calling RESIDIFF (1.0000), as is done in the constant velocity convection tests reprinted in Appendic B and discussed in the next section. This call can also be used to deliberately introduce some linear second-order diffusion to mimic the effect of viscosity. Since the nonlinear flux correction will leave additional dissipation in some regions, however, the user must remember that the residual diffusion may not be all that is present.

COPYGRID (MODE, I1, IN), with $MODE = 1$ is used to put aside a copy of all the grid variables generated by the last call to MAKEGRID. The arrays are copied into another named common block called OLD_GRID for subsequent reuse after some or all of these values may have been changed. Calling COPYGRID with $MODE = 2$ then subsequently restores all the values from cell $I1$ to cell IN including the first and last interfaces. While MAKEGRID could always be called to reset the grid, restoration from the copied values could save considerable computation – particularly if the computation of the interface locations, areas, or cell volumes is expensive. An example of where this is useful is a two-dimensional simulation where the grid is moving. Each of the NY rows needs to have MAKEGRID called before the half step and NEW_GRID (below) called to change the grid locations to those appropriate to the end of the timestep. COPYGRID could then be used to restore the grid to its condition before the most recent call to NEW_GRID and prior to beginning integration of the next row. It is also used in conjunction with SET_GRID to calculate cell volumes more efficiently when angular coordinates are being used.

NEW_GRID (RADR, I1, INP) performs the same function as MAKEGRID but is somewhat faster because the old values of the cell interface locations, $RADHO$, and all the variables which depend only on $RADHO$ are assumed unchanged since the last call to MAKEGRID. This means that a number of quantities do not have to be recalculated. This facility is useful, for example, when a moving grid at the end of a

complete timestep differs from the grid at the end of the half step but both integrations start from the same 'old' grid.

SET_GRID (FACTOR, I1, IN) is used when a multidimensional FCT model has one of the dimensions as an angular coordinate. In two dimenions it allows more efficient treatment of terms of the form

$$\frac{1}{r}\frac{\partial \rho}{\partial \theta} \tag{7.1}$$

where r and θ are the orthogonal radial and angular coordinates. For such a term, a cell area can be treated as a constant independent of radius, and a cell volume is proportional to radius, $\Lambda_i = r_j(\theta_{i+\frac{1}{2}} - \theta_{i-\frac{1}{2}})$, where r_j is the radius of the j^{th} row and $\theta_{i+\frac{1}{2}}$ is the angle of the i^{th} cell interface. In spherical coordinates the angle θ is replaced by $\zeta \equiv cos\theta$ with $\partial\zeta = sin\theta\partial\theta$ to obtain the equivalent 'Cartesian-like' coordinate. The integration in the θ direction is performed with $ALPHA$ set equal to one in the call to MAKEGRID. This call is located outside the loop over radius. Provisional cell volumes which are computed as the differences of the θ 's are stored in the arrays LOP and LNP and are contained in common block OLD_GRID when COPYGRID is called with $MODE = 1$. Inside the loop over radius SET_GRID must be called with the argument $RADR$ being the mean radius of the cell j and with $I1$ and IN being the first and last cells to be integrated. SET_GRID will multiply the $(\theta^{o,n}_{i+\frac{1}{2}} - \theta^{o,n}_{i-\frac{1}{2}})$ terms by the current cell radius r_j. This includes the radial effect in the cell volume calculation with only a single multiply for each row. This provides significant savings in the computational effort for obtaining cell volumes for each j^{th} row.

In Section 5 we discussed two routines called ZEROFLUX and ZERODIFF which modify the velocity dependent coefficients for certain specific boundary conditions where no fluxes or no diffusion across certain interfaces can be allowed on physical grounds. These requirements usually apply at the ends of the computational domain but there are circumstances, for example at a Lagrangian interface between two different materials or phases, where no flux of mass can be allowed in the frame of reference moving with the fluid. In this case ZEROFLUX (Interface) is called where 'Interface' is the Index of the interface separating the two phases, for example air and water. ZEROFLUX is also used at ideal solid walls to ensure that absolutely no flux into or out of the wall is allowed. By calling ZEROFLUX just before a particular continuity equation is solved, fluxers through the interface are permitted for all of the previously integrated equations.

ZERODIFF was designed to treat a slightly different problem: it turns off the diffusion and antidiffusion fluxes at chosen cell interfaces without affecting the convective fluxes. This can be used where there is a physically correct convective flux but where the numerical diffusion and/or antidiffusion would result in non-physical behavior. This routine should be

used for supersonic outflow conditions where the diffusion (or antidiffusion) might otherwise move some mass, momentum, or energy upstream in the flow. At inflows where the incoming fluxes are known, the guard cell values may not be exactly equal to the computed values just inside the mesh. There ZERODIFF is also used to ensure that the diffusion and antidiffusion terms can not change the prescribed fluxes entering the system. Examples using these routines are seen in the test problems and indicated in Tables 5.6–5.8.

As indicated repeatedly in the text, the LCPFCT routines rely on interface averaged quantities as inputs in several places. There is a great deal of freedom in computing these averages from the cell values on either side of the interface. In calculating the interface velocities, for example, the simple calculation,

$$v_{i-\frac{1}{2}} \equiv \frac{v_i + v_{i-1}}{2}, \tag{7.2}$$

could as well be computed as

$$v_{i-\frac{1}{2}} \equiv \frac{\rho_i v_i + \rho_{i-1} v_{i-1}}{\rho_i + \rho_{i-1}}. \tag{7.3}$$

This density-weighted average in Eq. (7.3) is equivalent to calculating the velocity at the interface from the average momentum divided by the average density.

There are infinitely many choices for these averages and there seems to be no compelling reason why $\{\rho_{i-\frac{1}{2}}\}$, $\{v_{i-\frac{1}{2}}\}$, $\{P_{i-\frac{1}{2}}\}$, and $\{(Pv)_{i-\frac{1}{2}}\}$ all have to be computed the same way. In Appendix C, Subroutine GASDYN performs these averages for LCPFCT Tests #2, #3, and #4 in yet another way. In the **Do 200** loop, an inverse density weighting,

$$P_{i-\frac{1}{2}} \equiv \frac{\rho_i P_{i-1} + \rho_{i-1} P_i}{\rho_i + \rho_{i-1}}, \tag{7.4}$$

is used not only for the pressure at the interfaces but also for the velocities and the pressure–velocity product. This is physically equivalent to calculating the pressure at the interface from the average of the squares of the sound speed in the two adjacent cells. The higher the sound speed the faster pressure is communicated so this choice in Eq. (7.4) makes as much physical sense as the simple average in Eq.(7.2) or the density weighted average in Eq. (7.3). In GASDYN the equations implementing the simple average of Eq. (7.2) are also included as comments following the **Do 200** loop in case the user wants to do a little experimenting.

In Subroutine VELOCITY there is also appreciable flexibility possible in choosing $\{\nu_{i-\frac{1}{2}}\}$ and $\{\mu_{i-\frac{1}{2}}\}$, the diffusion and antidiffusion coefficients $NULH$ and $MULH$. The calculation involving $SCRH$ is not strictly necessary but seems to improve the quality of the solutions of the convection test problem (Appendix B) a little without degrading

the quality of the other three test problem solutions. $SCRH$ subtracts a very small fourth-order velocity correction from the diffusion and then increases both the diffusion and antidiffusion by the same small second order amount. This latter correction has no effect on the residual diffusion of the algorithm but changes the pahse errors slightly, improving and symmetrizing the solutions for all three convection profiles. By setting $SCRH$ identically to zero, the user can verify that the absolute error measures of the convection problems do indeed increase.

In addition to the straightforward application of the LCPFCT routines, there are some tricks which help to give better solutions when the numerics are being stressed very hard. One of these is the use of minimum values. It is generally useful if the pressure is limited from below to be no smaller than some minimum value in computing the source terms for the momentum and energy equations in gas dynamic problems. This value is generally chosen to be one order of magnitude less than the smallest pressure that is physically expected. This prevents extreme undershoots in pressure occuringin problems involving strong shock waves. Also, when computing the velocity by dividing the fluid momentum by the density it is sometimes helpful to limit the value of the density used in the divisor to some minimum value. These ad hoc fixes are helpful because while the primary variables being integration by LCPFCT are guaranteed to be monotonic, the derived quantities such as velocity and pressure may not be monotone in the vicinity of steep gradients such as occur naturally at shocks. The effect of the use of these minimum values is highly localized since global conservation of the conserved quantities is still being maintained.

8. TEST PROBLEMS

In this section we describe four test problems that illustrate the application and capabilities of LCPFCT and provide benchmarks of the algorithms. A complete listing of the LCPFCT subroutines is given in Appendix A. The four test programs and selected reference outputs are given in Appendices B, C, D, and E.

1. The first test program, Appendix B, shows tests of linear convection with a fixed uniform velocity. Three different density profiles are convected, a discontinuous square wave profile, a semicircular profile, and a Gaussian peak profile. The program uses either the LCPFCT routine or the CNVFCT routine for this test; they are identical in the limit that the velocity field is constant.

2. The second test program propagates an ideal, one-dimensional, Rankine-Hugoniot shock through a gas with $\gamma = 1.4$. This program, listed in Appendix C, demonstrates the coupling of several continuity equations to solve a standard gasdynamic problem.

3. The third test program, Appendix D, simulates a one-dimensional exploding diaphragm (in a shock tube). This problem shows another example of gasdynamics with different boundary conditions and illustrates FCT's dynamic grid features.

4. The fourth test program is two dimensional and simulates the flow following the bursting of a diaphragm halfway up a finite length barrel. This program, in Appendix E, demonstrates the time splitting multidimensional implementation of LCPFCT on a geometrically complex domain and uses a simple extrapolated outflow condition.

For each test problem, we provide the driver program and selected printed outputs to five significant digits. This provides the basis for a fairly complete detection of any errors which might creep as the program is transferred, copied, or due to any Fortran or computer system incompatibilities. The five-digit printout comparisons should typically be adequate for validation on any computer with 32-bit or greater accuracy.

8.1 Constant Velocity Convection – LCPFCT Test #1

The first test program convects three different density profiles across a uniform grid at a constant velocity with periodic boundary conditions. The three density profiles chosen are a square wave (top-hat profile) 20 cells across, a semicircular density profile with a radius (called $WIDTH$) of 10 cells, and a Gaussian peak with a characteristic half width of five cells. These are all standard profiles for comparing numerical convection algorithms. The system length, NX, is 50 cells, the constant convection velocity, $VELX = 1.0$, and the calculation is carried out for $MAXSTP = 500$ timesteps with $DT = 0.2$ and $DX = 1.0$. Thus the nondimensional Courant number, $\epsilon \equiv VELX \times DT/DX = 0.2$, is 40% of the maximum allowed by this FCT algorithm. The standard run is long enough for

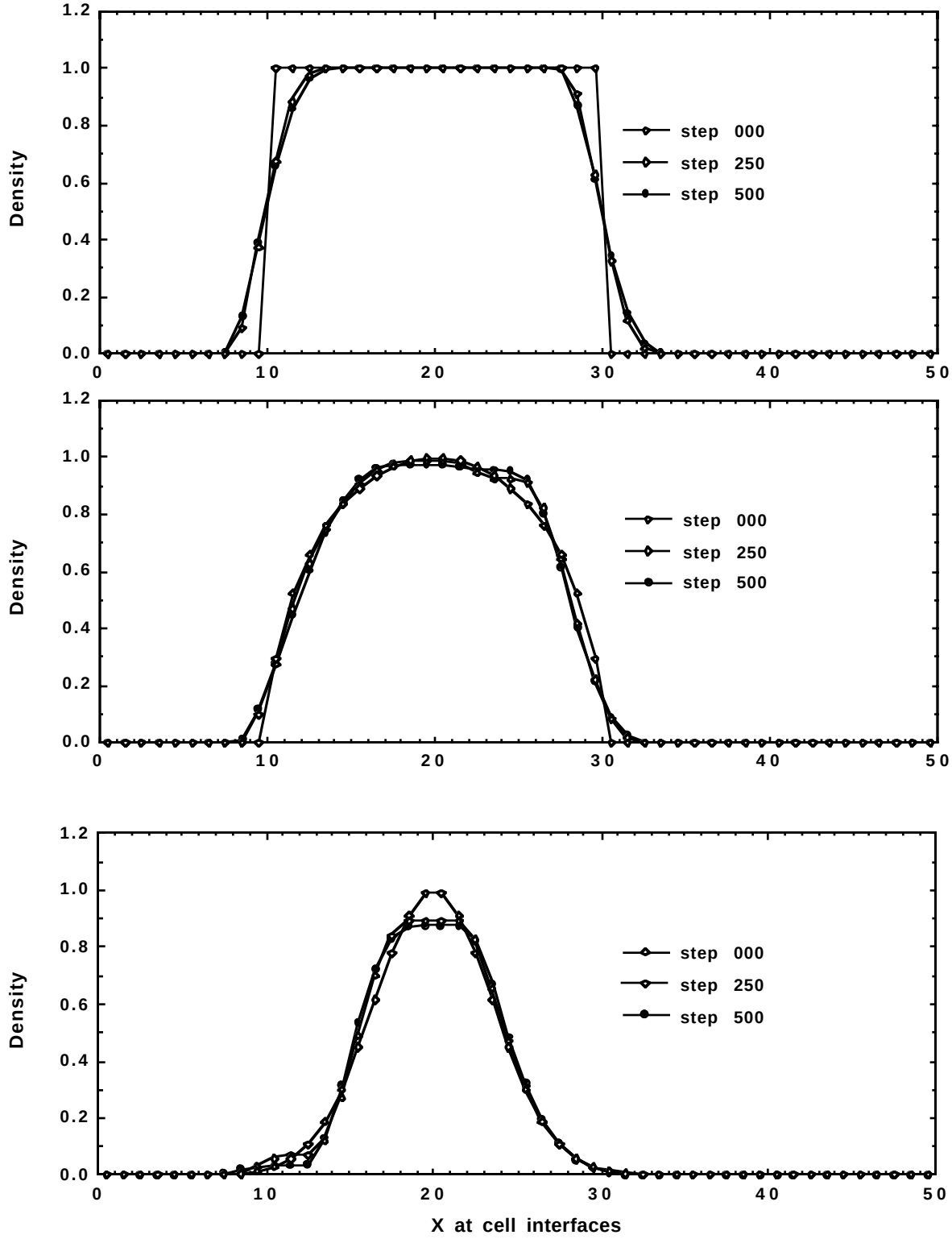

Figure 8.1 Results of the convection tests with LCPFCT on a) square wave profile, b) circular arch profile, and c) gaussian peak profile after 250 and 500 timesteps with a Courant number of 0.2.

the density profiles to travel across the mesh twice, coming back to the initial position each time because periodic boundary conditions are used. This type of calculation with periodic boundary conditions is a good test of the algorithm's numerical fidelity because the successive profiles should exactly overlay each other.

Listings of the program used to generate these results along with the printed output at steps 125 and 500 are given in Appendix B. The printed solutions labeled as 'exact' at step 500 are also the initial conditions for each of the profiles. These results for the three different test profiles are also shown in Figure 8.1 for steps 250 and 500 overlaid on the initial conditions. The quantities marked 'absolute error' at the bottom of the printouts are the sum of the absolute values of the differences between the computed solution and the exact solution normalized by the conserved sum of the density values for each of the profiles. The exact (analytic) solution is found by following a translating grid on which the solution remains exact and performing a simple numerical integration to get the correct average of the chosen profile over each cell.

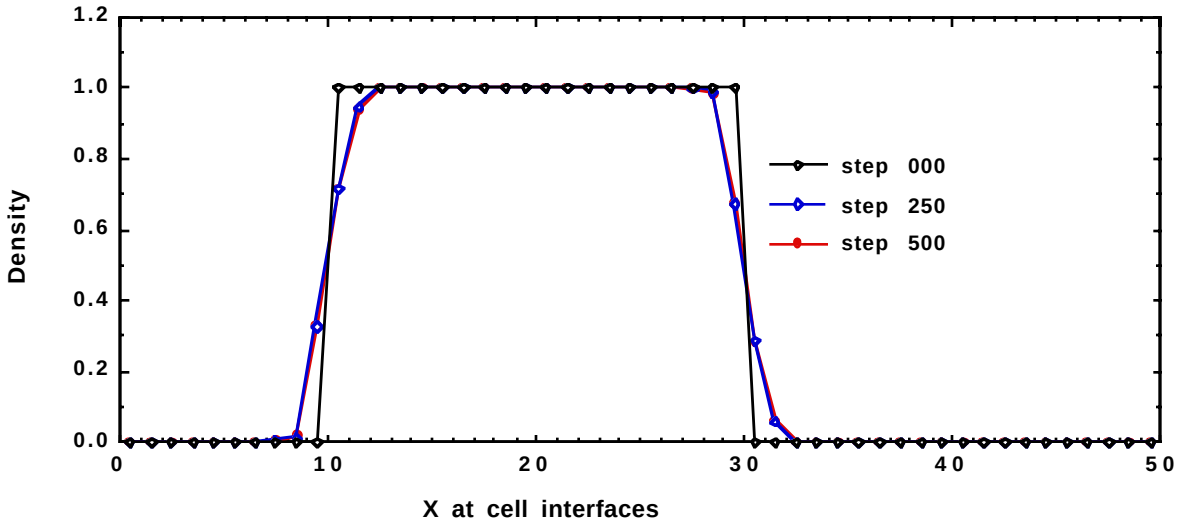

Figure 8.2 Convecting the square wave profile using LCPFCT with enhanced antidiffusion. The values of the coefficients $MULH$ in common block FCT_VELO have been increased 2%. The density profile after 250 and 500 timesteps of convection with a Courant number of 0.2 is overlaid on the initial conditions.

A linear convection algorithm has amplitude, phase, and Gibbs errors which cause the numerical solution to deviate from the expected theoretical solution. Since FCT is intrinsically nonlinear, these three types of errors are mixed together, reappearing as profile 'rounding,' 'terracing,' and 'clipping.' The solutions shown in Figure 8.1 illustrate these three different types of error. Because of the particular test profiles chosen, each of these types of error is best illustrated by a different profiles. The corners of the square wave are

'rounded' somewhat to generate a modified profile that propagates essentially unchanged. In the semicircular profile, a small 'terrace' forms around cell 25 where phase errors deform the shape of the convected profile somewhat until a new maximum begins to appear where the slope is relatively small. This new extremum is prevented by the antidiffusive flux limiter, leading to the terrace. In the gaussian peak problem, the rather sharp peak is 'clipped' off. The diffusion stage reduces the height of the two highest points and then the flux corrector prevents the antidiffusion flux from reconstituting these peaks.

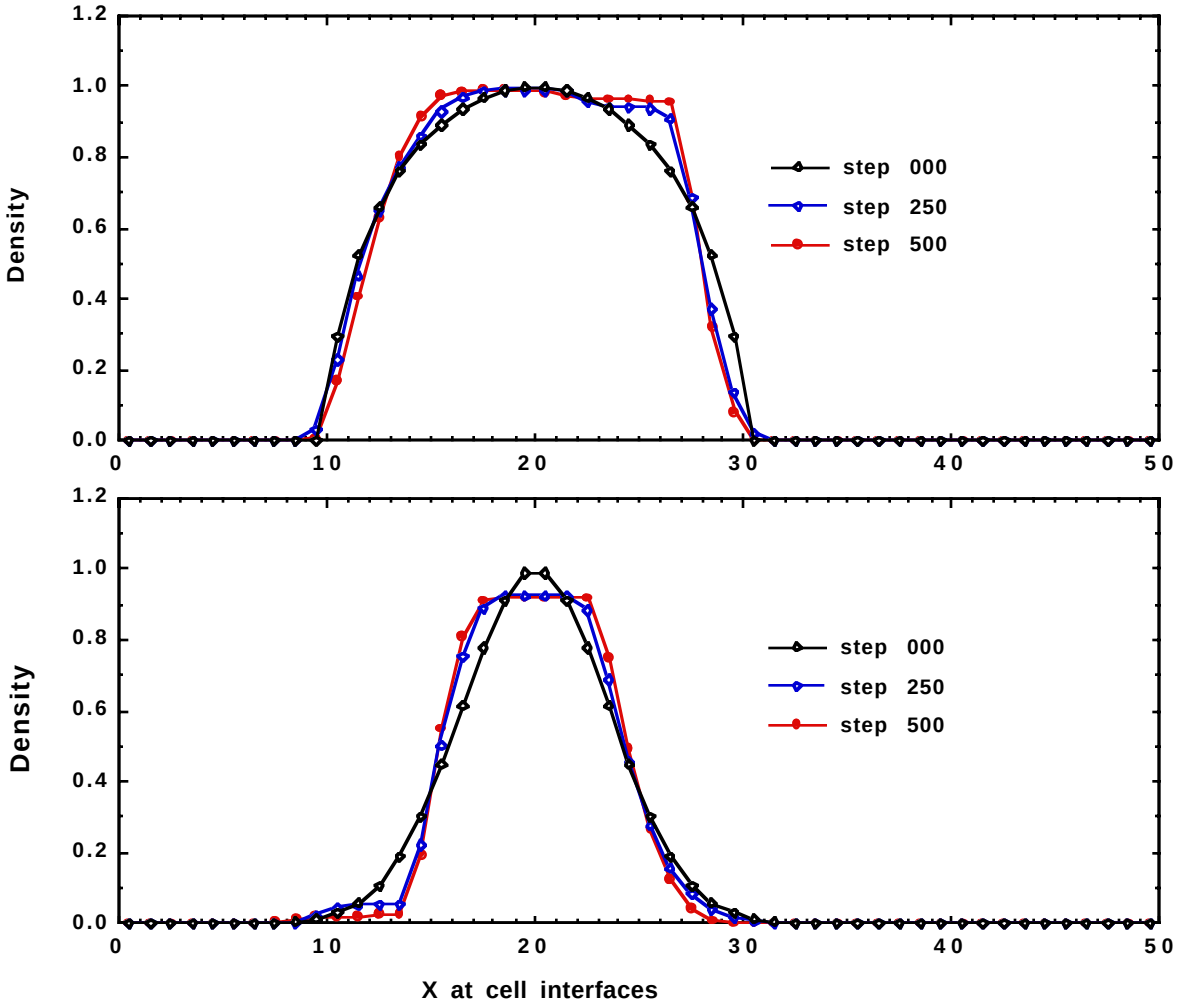

Figure 8.3 Results of the convection tests with LCPFCT on the, a) semicircular density profile, and the b) gaussian peak density profile after 250 and 500 timesteps with a Courant number of 0.2. The values of the coefficients $MULH$ in common block FCT_VELO have been increased 2%.

It is difficult to do anything about the phase errors at short wavelength which lead, in particular, to the terracing effect but it is tempting to try to reduce the rounding and

clipping by modifying with the amount and conditions of the flux-correction procedure. For example, the antidiffusion coefficients in the LCPFCT algorithm can be increased slightly by multiplying the $NX+1$ values of $\mu_{i+1/2}$ $(MULH)$ by a value slightly larger than unity. This makes the linear algorithm slightly unstable but improves the fidelity of physically discontinuous or nearly discontinuous solutions significantly, as seen in Figure 8.2 for the square wave profile. In Figure 8.2 the number of zones in the numerically approximated discontinuity is 2 or 3 while it is 4 or 5 in Figure 8.1a. Further there is virtually no additional rounding evident between steps 250 and 500 in Figure 8.2.

Some monotone methods employ 'contact surface steepeners' such as just described in regions of flow where contact surfaces can be identified. These techniques should be used with caution, however, because their long term numerical stability depends totally on the nonlinear action of the flux corrector. Figure 8.3 shows the results of using this approach on the second and third profiles which are not as discontinuous as the square wave. It can be seen that enhancing the antidiffusion trys to turn even smooth profiles into square waves. The terracing is enhanced significantly in Figure 8.3a and the clipping is much more extreme in Figure 8.3b. The bases of the profiles are pulled in and the tops broadened out while still conserving the total integral under the curve and the mean position of the fluid.

8.2 Progressing One Dimensional Gasdynamic Shock – LCPFCT Test #2

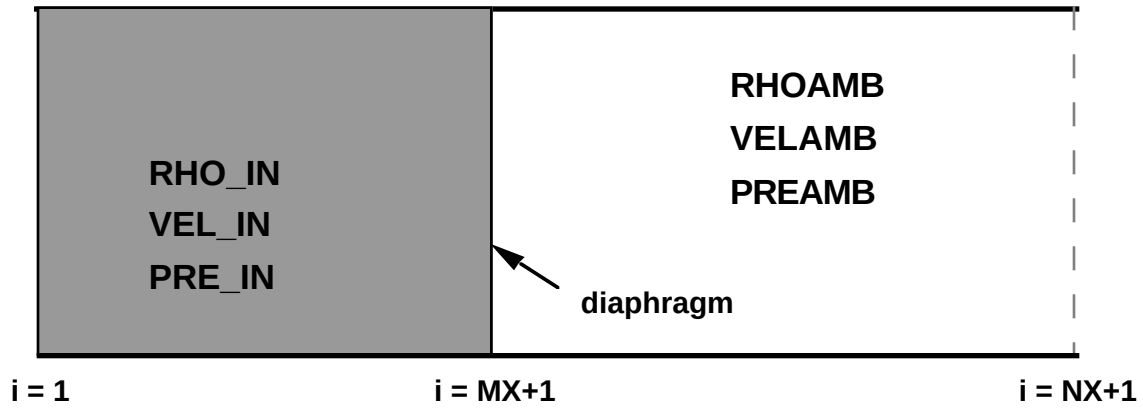

Figure 8.4 Schematic diagram showing the one-dimensional geometry and key initial parameters for the Progressing Shock Test and the Bursting Diaphragm Test.

In one dimension the equations of ideal gasdynamics can be written as three nonlinearly coupled continuity equations. In Section 4 we showed how LCPFCT can be used to solve each of these continuity equations through a structured sequence of calls. Appendix C, which contains the Fortran program conducting this progressing shock test, also includes a subroutine called GASDYN which captures this series of calls to LCPFCT and its sup-

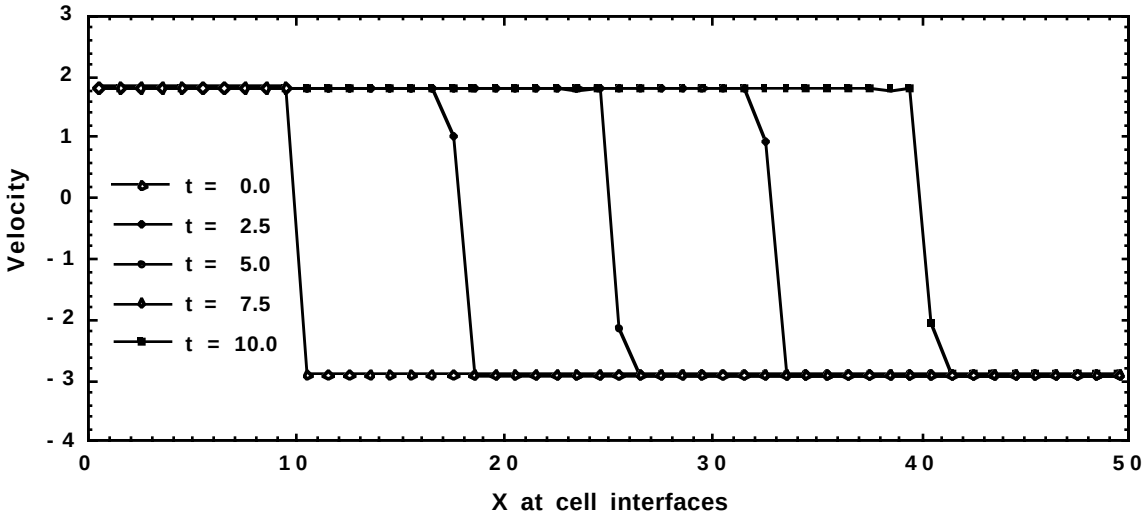

Figure 8.5 Velocity profiles at intervals of 50 timesteps from the LCPFCT Progressing Shock Test – Problem #2. The shock velocity, V0, was chosen as 3.0 so the shock moves 7.5 cells in each interval.

porting routines. GASDYN can be used in a number of circumstances including Tests #3 and #4 below.

Figure 8.4 shows the initial configuration for a progressing shock problem. The shock moves in the positive X direction with a velocity $V0 = 3.0$. The fluid, entering the system on the right at velocity $VELAMB = -2.9161$, is a Mach 5 flow relative to the oncoming shock. The ambient (preshock) density is $RHOAMB = 1.0$ and the corresponding pressure is $PREAMB = 1.0$. Inside the shocked region, initialized to the left of interface 11 in this case, the fluid conditions are $VEL\_IN = 1.8168$, $RHO\_IN = 5.0$, and $PRE\_IN = 29.0$. In this progressing shock test program and the one for the bursting diaphragm discussed below, a fourth variable $RVTN$, denoting an unused transverse momentum, is initialized to zero. This variable is important in two-dimensional flows and allows GASDYN to be used as well for Test #4, the two-dimensional "muzzle flash" problem. The boundary conditions chosen are of type 4 for the calls to GASDYN, meaning that fixed, known values are enforced on the guard cell variables and thus allowed to enter the computational domain. The system is $NX = 50$ cells long and $DELTAX = 1.0$. The timestep $DELTAT$ is fixed at 0.05.

The ideal propagating shock is self steepening so its computation using LCPFCT reaches a limiting profile as it moves across the grid. This profile is physically correct and numerically stable though it is not completely steady in time. Since the location of the shock relative to the cell center changes from one step to the next, the profile changes slightly. Small fluctuations arise as the shock progresses from one cell to the next,

depending on the one or two cells involved in the transition from the pre-shock to the post-shock conditions and exactly where in the cell the shock resides.

The outputs reproduced in Appendix C, step 0 ($t = 0.0$), step 150 ($t = 7.5$), and step 200 ($t = 10.0$), show that the pressure and velocity have small nonmonotone undershoots of order 0.1% just in front of the shock. This error is so small that the effect cannot be seen in Figure 8.5. It occurs because the shock is so steep that the energy density and momentum density transitions only have to get a little bit out of phase for quantities derived from two or more of the primary conserved variables, like the pressure or the velocity, to show structure not seen in the primary variables themselves. There has been a substantial amout of work over the years concerned with such issues for algorithms where the effects are orders of magnitude worse. In most circumstances where the unwanted effects are very small, as in the cases here, the cure can be worse than the disease.

8.3 One-Dimensional Bursting Diaphragm Problem – LCPFCT Test #3

The bursting diaphragm test, also illustrated in Figure 8.4, solves the ideal one-dimensional equations of gas dynamics for the classical shock tube problem: an initial condition in which two uniform perfect gases at rest are separated by a diaphragm which breaks cleanly and instantaneously at time zero. The region to the left of the interface at $MX + 1 = 61$ consists of a gas at 10 times the pressure on the right but at the same density. That is, $RHOAMB = RHO\_IN = PREAMB = 1.0$, $PRE\_IN = 10.0$, and $VELAMB = VEL\_IN = 0.0$. The system has $NX = 100$ cells with $DELTAX = 1.0$ initially. In this test, the grid is adaptively changed in time as described below.

The boundary conditions chosen are of type 1 for the call to GASDYN specifying ideal hard reflecting end walls. Since the gridding is such that the solution does not reach either end of the computational domain and the external solutions on the right and left are constant states, this choice is not unique or even important. The timestep was held fixed at $DELTAT = 0.05$ because the grid variations are relatively small. Tabulated results are printed at steps 0, 100, 200, and 1600 in Appendix D. These include times before and after the grid stretching is started and show the approach to the expected theoretical similarity solution. Listings of the test program are also given in Appendix D.

Figure 8.6a shows the computed density profiles ($RHON$) at times 5.0 (step 100), 7.5 (step 150), and 10.0 (step 200) before the grid has begun to expand adaptively with the solution. This initial transient has four flow characteristic surfaces forming from the single diaphragm discontinuity and separating. They bound three distinct broadening regions which only become well resolved after some time. In the figure the rightmost region, lead by a shock and followed by a contact surface, moves off to the right and is centered around X = 20 at time 10.0 when the grid stretching is begun. Figure 8.6b shows four of the

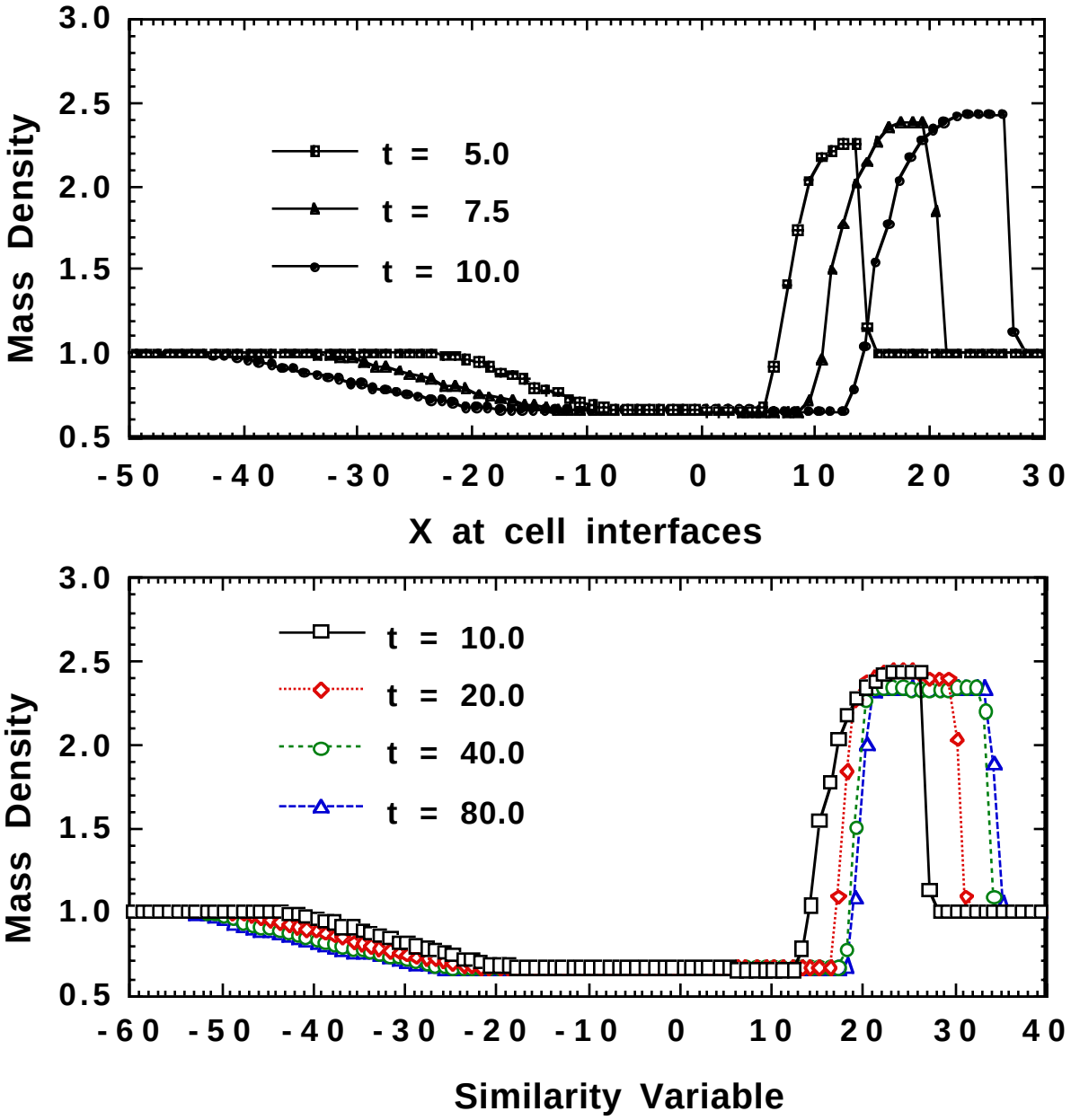

Figure 8.6 Evolution of the mass density profile for the bursting diaphragm problem, LCPFCT Test #3. a) Initial transient evolution on a 'fixed' grid. b) Evolution on an expanding grid showing approach to the analytic similarity solution.

density profiles at times 10.0 (step 200), 20.0 (step 400), 40.0 (step 800), and 80.0 (step 1600). At the beginning of this sequence (step 200), the grid begins to linearly stretch at a rate of $V/X = .0775$ which causes the various characteristic surfaces in the flow to come to rest in the stretching grid. A look at the program in Appendix D will show how this is implemented using MAKEGRID.

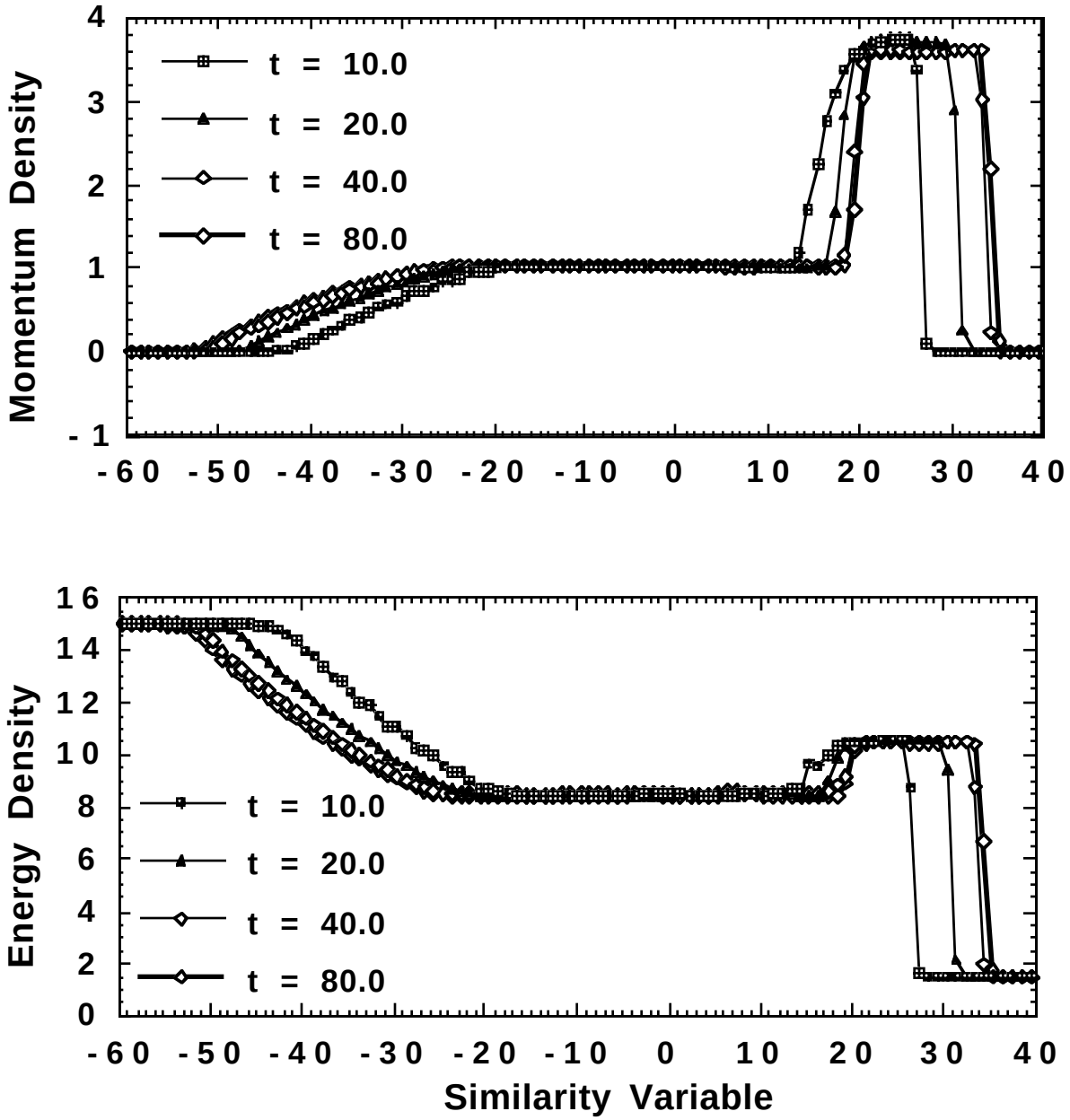

Figure 8.7 a) X momentum density ($RVXN$) and b) energy density ($ERGN$) shown at timesteps 200, 400, 800 and 1600 after grid expansion has begun. The horizontal axis is the similarity variable acheived by expanding the grid.

Figure 8.7 shows that same convergence of four computed solutions toward the expected similarity solution for the momentum density ($RVXN$) and the energy density ($ERGN$). The terracing exhibited by the early solutions in the rarefaction wave region to the left of the initial location of the diaphragm has almost disappeared as the solution progresses. Since the grid was already being moved in this test, we superimposed a small oscillation from one step to the next whose amplitude is controlled by DX_OSC in the

program, here chosen rather arbitrarily as 0.125. This flexibility to move the cells around rather arbitrarily, as long as interfaces don't cross each other, is very useful in Lagrangian, quasi-Lagrangian, and adaptively gridded "sliding rezone" calculations.

8.4 Two-Dimensional Muzzle Flash Problem – LCPFCT Test #4

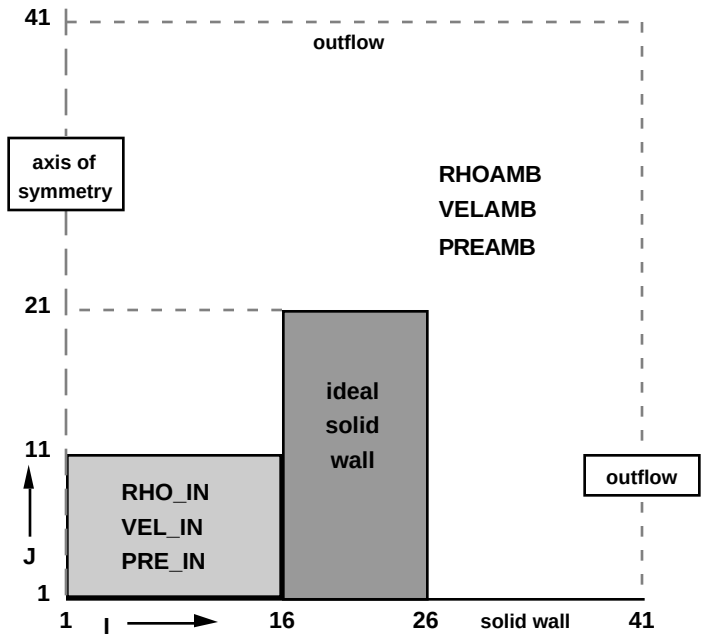

Figure 8.8 Diagram of the geometry and initial conditions of the two-dimensional muzzle flash problem – LCPFCT Test #4. The cylindrical wall ("muzzle") is 1.0 cm thick and 2.0 cm long. The pressure ratio behind the diaphragm is 1000:1 and the density ratio is 100:1.

Here we show how timestep splitting is used to solve the ideal two-dimensional equations of gasdynamics with the LCPFCT modules. This listing in Appendix E shows how to set up the calls to the grid subroutine and impress boundary conditions for a diaphragm problem like Test #3 but this time in a cylindrical barrel. Figure 8.8 shows the geometric configuration and dimensions of the distinct regions in this problem. After the shock has formed just above the diaphragm at interface $J = 11$, it progresses for a short distance in a restricted one-dimensional manner and then flashes out of the muzzle and expands in an axisymmetric manner. Simple outflow boundary conditions, BC_OUTF, are applied to the upper and right boundaries at interfaces $J = 41$ and $I = 41$ respectively. The lower boundary at interface $J = 1$ and the axis at interface $I = 1$ were solved with reflecting boundary conditions, BC_WALL and BC_AXIS respectively. The calculation is performed on a 40×40 uniformly spaced rectangular grid with $DR = DZ = 0.1$. The equations are not integrated inside the ideal solid cylindrical wall shown in Figure 8.6.

The initial conditions were formed by setting the entire mesh to an ambient values, $RHOAMB = 0.00129$, $VELAMB = 0.0$, and $PREAMB = 1.013 \times 10^6$, for the density, velocity and pressure. We then created a region 10 cells high inside the barrel and below the diaphragm with $RHO\_IN = 0.129$ and $PRE\_IN = 1.103 \times 10^9$, one hundred and one thousand times ambient respectively. The fluid everywhere was assumed to be a perfect gas with $\gamma = 1.4$.

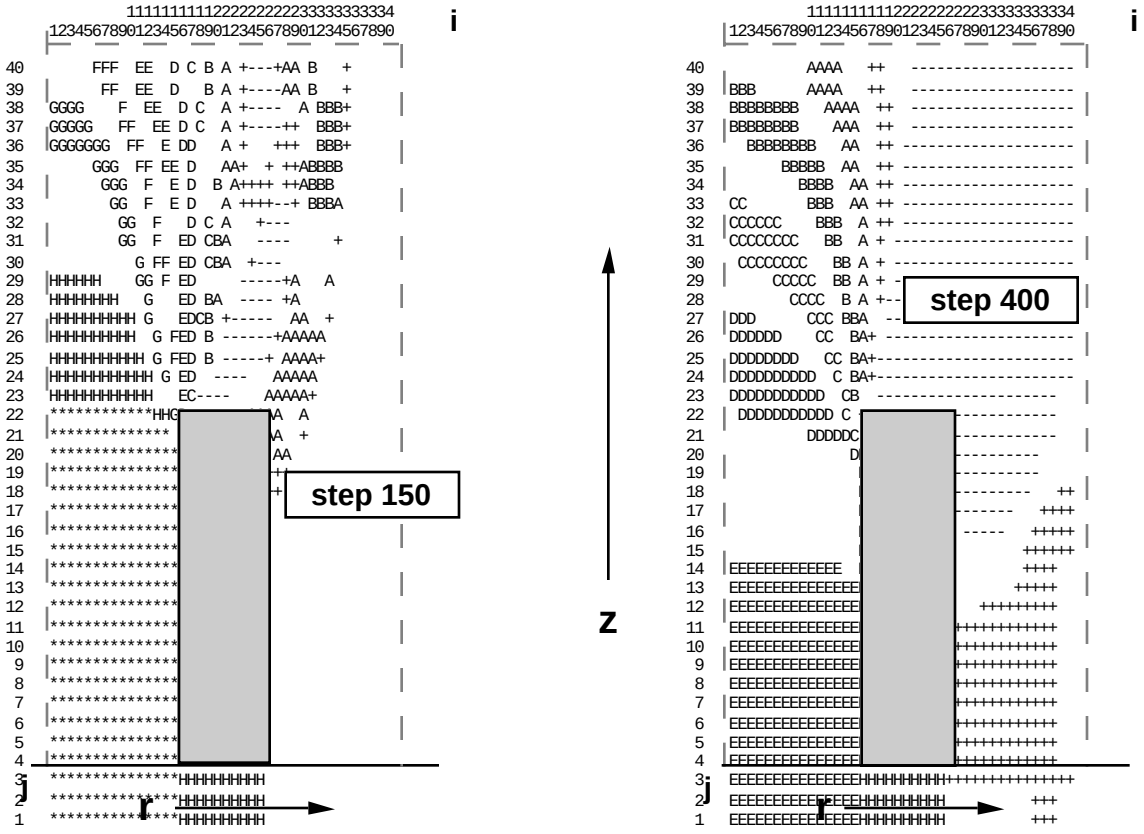

Figure 8.9 Character contour plots of the solution to the two-dimensional muzzle flash problem at two different times a) step 150, and b) step 400.

The calculation is performed in R–Z cylindrical geometry by specifying $IALFR = 2$ rather than 1 in the appropriate call to MAKEGRID for integrations in the radial direction. In the way LCPFCT is implemented, the first row and first column are one half zone from the reflecting boundaries. This facilitates conservation checks and simplifies application of the boundary conditions since the the first row and column represent a full cell and conservation can be enforced by requiring the fluxes through the boundaries to be zero. Results are shown as printouts of two lines through the two dimensional arrays at steps 150 and 400. A simple variable timestep capability is included with this test to show one easy

way to do this. The square root of the energy divided by the density gives a reasonable approximation to the fastest characteristic speed in the problem. Therefore the maximum of this speed over the grid is used to estimate a suitable timestep for the next step of integration.

The listing of the driver program and the printed results are in Appendix E. Figure 8.9 shows two simple character contour plots of the density computed by this model for timesteps 150 and 400 at times 25.06μsec and 75.06μsec of the calculation respectively. In Figure 8.9a the flow has just left the muzzle and is reaching the upper boundary. This shows flow exiting the computational domain with no appreciable numerical reflection through the implementation of boundary condition 2. At the later time of Figure 8.9b, the extrapolated outflow conditions at the radial boundary has also come into play and is showing no reflection of a purely numerical origin. This simple outflow boundary condition is implemented in subroutine GASDYN.

Since the GASDYN subroutine provided for the earlier LCPFCT Test #2 and Test #3 included provision for a second transverse momentum $RVTN$, it can be used as well for this two-dimensional problem. In a production environment, it is best to have different versions of GASDYN to reduce the confusion in the programming and unnecessary statements and tests in the Fortran related to directions and variables not being used. Here however, using the same code reduces the number of different programs while showing just how straightforward this approach to fluid dynamics is. The extension of this test program and the subroutine GASDYN to three dimensions is also a rather easy modification. Indeed, the simplicity and efficiency of FCT algorithms has led to their ready adaptation to efficient parallel processing, e.g., Gustafson et al. (1988) and Oran and Boris (1991).

Since FCT is designed to solve general continuity equations, complex fluid dynamic interactions involving rotational flow and turbulence can be simulated as easily as the transient shock and blast problems used as examples here. Furthermore, extensive tests performed with FCT and monotone algorithms, for example by Edgar and Woodward (1991), and Boris et al. (1992), indicate that these methods behave as large eddy models are expected to behave for high Reynolds number flows. The flux-correction procedure adds a local, solution-dependent dissipation to the overall fluid dynamic simulation much as an eddy viscosity (diffusivity) would, coverting nonlinearly generated, unresolvable structure in the computed flow to heat, modeling the effect of a physical viscosity acting at the small scales of a turbulent cascade.

9. SUMMARY

This report has described a series of subroutines which, as a library, constitute the Fortran implementation of the LCPFCT algorithm. This version of the FCT algorithm is the culmination of two decades for research into FCT algorithms and represents the natural extension of the previous ETBFCT and PRBFCT subroutines. We have attempted to make these subroutines as general as possible without appreciably sacrificing simplicity or efficiency. The routines, as they stand, treat a wide range of geometries, grid configurations, boundary conditions, and source terms. "Hooks" have been left in these routines to add additional geometries and source terms and some of these advanced capabilities are discussed briefly in Section 7. The programs and documentation both indicate places where an advanced user may wish to make other changes such as employing different weighted averages for computing interface quantities from the cell-averaged conserved variables. This report describes only the low-phase error, one-dimensional FCT algorithm in the context of a two-stage Runge-Kutta time integration but this is the version which we have found easiest to implement on different computers including a number of parallel processing systems and easiest to teach to people who are just learning CFD and need to get good multidimensional results in a short period of time.

Here we would like to reiterate the philosophy behind the data and program structure adopted in LCPFCT. To solve Eq. (1.1) numerically for even one timestep requires a great deal of input data and many independent calculations. The logically separate components of this calculation, 1) the data structures used, 2) the geometry of the computational domain, 3) the system of continuity equations being solved, and 4) the algorithms used to solved each of these continuity equations, have been built into a hierarchical program with distinct subroutines requiring separate calls. This separation of function has two advantages. First, many different types of calculations can be performed using the same code by mixing and matching various functions (subroutines) in different orders and with different arguments. Second, functions and calculations which are common to large regions of the computational domain or to several different continuity equations or which are unchanged over several timesteps do not need to be performed more than once. Thus optimization is achieved by eliminating many repeated calculations even though a slight extra expense is sometimes incurred to construct extra calling sequences made necessary by the strict separation of function employed in the subroutines.

As a final note of caution, we point out that while the LCPFCT routines have been checked out carefully, by no means has every possible application been tested. Many combinations of calls and valid applications have not yet even been thought of. Thus there may be bugs which appear in new applications. This warning is particularly important because FCT algorithms are notoriously forgiving and uncomplaining. They seldom blow

up (unless $U\frac{\delta t}{\delta x} > \frac{1}{2}$), but rather the solution degrades gracefully and unspectacularly. This may sound like an advantage but it can be extremely misleading since many potential users may be accustomed to methods which fall apart quickly when there are even minor difficulties. Be skeptical; always check your numbers carefully, particularly near the boundaries. Do not just look at the exponents of the answers or the general (graphical) properties of your solutions.

Any suggestions for improvements, either for the subroutines or for this exposition will be gratefully received by the authors and valid changes incorporated in future editions/printings.

Acknowledgments

We would like to acknowledge the work and contributions of our many colleagues at NRL and elsewhere who have participated in the development of flux-corrected transport algorithms over the last two decades. David Book, C. Richard DeVore, K. Kailasanath, Fernando Grinstein, Raafat Guirguis, Chiping Li, B. Edward McDonald, Gopal Patnaik, Stephen Zalesak, and have contributed to the theoretical development and understanding of FCT algorithms, have invented new FCT algorithms, have undertaken extensive testing and analysis of the performance of various FCT algorithms, and have used these methods in the performance of large-scale CFD research for the Navy, DoD, and the United States.

This work involved in preparing this report has been supported by the Defense Advanced Research Projects Agency, the Office of Naval Research, and the Naval Research Laboratory. The development of the original Flux-Corrected Transport algorithms was supported by DNA, DOE and NRL.

REFERENCES

Baer, M.R., and R.J. Gross, 1986, A Two-dimensional Flux-Corrected Transport Solver for Convectively Dominated Flows, SAND85–0613, Sandia National Laboratories, Albuquerque, NM.

Book, D.L., and J.P. Boris, 1981, *Finite-Difference Techniques for Vectorized Fluid Dynamics Calculations*, Chapter 2, Springer-Verlag, New York.

Book, D.L., C. Li, G. Patnaik, and F.F. Grinstein, 1991, Quantifying Residual Numerical Diffusion in Flux-Corrected Transport Algorithms, *J. Sci. Comp.* **6** (3): 323-343.

Boris, J.P., 1971, A Fluid Transport Algorithm that Works, in *Computing as a Language of Physics*, International Atomic Energy Agency, Vienna, 171–189.

Boris, J.P., 1976, Flux-Corrected Transport Modules for Generalized Continuity Equations, NRL Memorandom Report 3237, Naval Research Laboratory, Washington, DC.

Boris, J.P., F.F. Grinstein, E.S. Oran, and R.L. Kolbe, 1992, New Insights into Large Eddy Simulation, to appear, *Fluid Dynamics Research*.

Boris, J.P., and D.L. Book, 1973, Flux-Corrected Transport I: SHASTA — A Fluid Transport Algorithm that Works, *J. Comp. Phys.* **11**: 38–69.

Boris, J.P., and D.L. Book, 1976, Solution of the Continuity Equation by the Method of Flux-Corrected Transport, *Methods in Computational Physics* **16**: 85–129.

Colella, P., and P.R. Woodward, 1984, The Piecewise Parabolic Method (PPM) for Gas-Dynamical Simulations, *J. Comp. Phys.* **54**: 174–201.

DeVore, C.R., 1989, Flux-Corrected Transport Algorithms for Two-Dimensional Compressible Magnetohydrodynamics, U.S. Naval Research Laboratory Memorandum Report 6544.

DeVore, C.R., 1991, Flux-Corrected Transport Techniques for Multidimensional Compressible Magnetohydrodynamics, *J. Comp. Phys.* **92**: 142–160.

Edgar, B.K., and P.R. Woodward, 1991, Diffraction of a Shock Wave by a Wedge: Comparison of PPM Simulations with Experiment, *AIAA Paper 92–0696*: 1–29.

Fyfe, D.E., and G. Patnaik, 1991, Parallel Implementation of Multi-Dimensional FCT on Non-Orthogonal Meshes, Proceedings, 4th International Symposium on Computational Fluid Dynamics, Davis CA.

Givoli, D., 1991, Non-Reflecting Boundary Conditions, *J. Comp. Phys.* **94**: 1–29.

Godunov, S.K., 1959, Finite Difference Methods for Numerical Computation of Discontinuous Solutions of the Equations of Fluid Dynamics, *Mat. Sb.* **47**: 271–306.

Grinstein, F.F., 1992, Open Boundary Conditions in the Simulation of Subsonic Turbulent Shear Flows, submitted to *J. Comp. Phys.*

Grinstein, F.F., and R.H. Guirguis, 1992, Effective Viscosity in the Simulation of Spatially Evolving Shear Flows with Monotonic FCT Models, *J. Comp. Phys.* **101**: 165–175.

Grosch, C.E., and S.A. Orszag, 1977, Numerical Solution of Problems in Unbounded Regions: Coordinate Transformations, *J. Comp. Phys.* **25**: 273–295.

Gustafson, J.L., G.R. Montry, and R.E. Benner, 1988, Development of Parallel Methods for a 1024-Processor Hypercube, *SIAM J. Sci. Stat. Comp.* **9**: 609–638.

Harten, A., 1974, *The Method of Artificial Compression*, CIMS Rept. COO-3077-50, Courant Institute, New York University, New York.

Harten, A., 1983, High Resolution Schemes for Hyperbolic Conservation Laws, *J. Comp. Phys.* **49**: 357–93.

Kutler, P. (ed), 1982, *Numerical Boundary Condition Procedures*, NASA Conference Publication 2201, NASA Ames Research Center, Moffett Field, CA.

Lafon, F., and S. Osher, 1992, Essentially Nonoscillatory Postprocessing Filtering Methods, NASA Contractor Report 189610, ICASE–92–05.

Landsberg, A.M., and J.P. Boris, 1992, An Efficient Method for Solving Flows Around Complex Bodies, U.S. Naval Research Laboratory Memorandum Report, to appear.

Lax, P.D., and B. Wendroff, 1964, Difference Schemes for Hyperbolic Equations with High Order of Accuracy, *Comm. Pure Appl. Math.* **17**: 381–398.

Löhner, R., K. Morgan, J. Peraire, and M. Vahdati, 1987, Finite Element Flux-Corrected Transport (FEM–FCT) for the Euler and Navier-Stokes Equations, *Int. J. Num. Meth. Fluids* **7**: 1093–1109.

Leonard, B.P., and H.S. Niknafs, 1990, A Cost Effective Strategy for Nonoscillatory Convection Without Clipping, NASA Technocal Memorandum 102538, ICOMP–90–09.

Nessyahu, H., and E. Tadmor, 1990, Non-Oscillatory Central Differencing for Hyperbolic Conservation Laws, *J. Comp. Phys.* **87**: 408–463.

Odstrcil, D., 1990, A New Optimized FCT Algorithm for Shock Wave Problems, *J. Comp. Phys.* **91**: 71–93.

Oran, E.S., and J.P. Boris, 1987, *Numerical Simulation of Reactive Flow*, Chapter 8, Elsevier, New York.

Oran, E.S. and J.P. Boris, 1991, Compressible Flow Simulations on a Massively Parallel Computer, *International Journal of Modern Physics C*, 430–436.

Patnaik, G., R.H. Guirguis, J.P. Boris, and E.S. Oran, 1987, A Barely Implicit Correction for Flux-Corrected Transport, *J. Comp. Phys.* **71**: 1–20.

Poinsot, T.J. and Lele, S.K., 1992, Boundary Conditions for Direct Simulations of Compressible Viscous Flows, *J. Comp. Phys.* **101**: 104–129.

Rood, R.B., 1987, Numerical Advection Algorithms and their Role in Atmospheric Transport and Chemistry Models, *Rev. Geophys.* **25**: 71–100.

Thompson, K.W., 1987, Time Dependent Boundary Conditions for Hyperbolic Systems, *J. Comp. Phys.* **68**: 1–24.

Thompson, K.W., 1990, Time Dependent Boundary Conditions for Hyperbolic Systems, II, *J. Comp. Phys.* **89**: 439–461.

Turkel, E., 1980, Numerical Methods for Large-Scale Time-Dependent Partial Differential Equations, in W. Kollmann, ed., *Computational Fluid Dynamics, Volume 2*, Hemisphere, Washington DC, 127–262..

van Leer, B., 1973, Towards the Ultimate Conservative Difference Scheme. I. The Quest of Monotonicity, in H. Cabannes and R. Temam, eds., *Lecture Notes in Physics* **18**, Springer-Verlag, Berlin, 163–168.

van Leer, B., 1979, Towards the Ultimate Conservative Difference Scheme. V. A Second Order Sequel to Godunov's Method, *J. Comp. Phys.* **32**: 101–136, 1979.

Woodward, P., and P. Colella, 1984, The Numerical Simulation of Two-Dimensional Fluid Flow with Strong Shocks, *J. Comp. Phys.* **54**: 115–173.

Zalesak, S.T., 1979, Fully Multidimensional Flux-Corrected Transport Algorithms for Fluids, *J. Comp. Phys.* **31**: 335–362.

Zalesak, S.T., 1981, Very High Order and pseudospectral Flux-Corrected Transport (FCT) Algorithms for Conservations Laws, in R. Vichnevetsky and R.S. Stepleman, *Advances in Computer Methods for Partial Differential Equations*, Vol. IV, 126–134.

APPENDIX A. LISTING OF LCPFCT LIBRARY SUBROUTINES

The thirteen subroutines of the LCPFCT library are listed sequentially in the following pages. The four main subroutines, LCPFCT, MAKEGRID, VELOCITY, and SOURCES are listed first followed by the remaining routines in alphabetic order. The following index is provided to direct the reader to the page where the listing of each of the routines, including the block data routine FCTBLK, begins.

```
C=====================================================================

      Subroutine LCPFCT ( RHOO, RHON, I1, IN,
     &                    SRHO1, VRHO1, SRHON, VRHON, PBC )

C---------------------------------------------------------------------
c
c     Originated: J.P. Boris         Code 4400, NRL          Feb 1987
c     Modified:  Laboratory for Computational Physics & Fluid Dynamics
c     Contact:    J.P. Boris, J.H. Gardner, A.M. Landsberg, or E.S. Oran
c
c     Description:  This routine solves generalized continuity equations
c     of the form  dRHO/dt = -div (RHO*V) + SOURCES in the user's choice
c     of Cartesian, cylindrical, or spherical coordinate systems.  A
c     facility is included to allow definition of other coordinates.
c     The grid can be Eulerian, sliding rezone, or Lagrangian and can
c     be arbitrarily spaced.  The algorithm is a low-phase-error FCT
c     algorithm, vectorized and optimized for a combination of speed and
c     flexibility.  A complete description appears in the NRL Memorandum
c     Report (1992), "LCPFCT - A Flux-Corrected Transport Algorithm For
c     Solving Generalized Continuity Equations".
c
c     Arguments:
c     RHOO   Real Array        grid point densities at start of step   I
c     RHON   Real Array        grid point densities at end of step     O
c     I1     Integer           first grid point of integration         I
c     IN     Integer           last grid point of intergration         I
c     SRHO1  Real Array        boundary guard cell factor at cell I1+1 I
c     VRHO1  Real Array        boundary value added to guard cell I1-1 I
c     SRHON  Real Array        boundary guard cell factor at cell IN+1 I
c     VRHON  Real Array        boundary value added to guard cell IN+1 I
c     PBC    Logical           periodic boundaries if PBC = .true.     I
c
c        In this routine the last interface at RADHN(INP) is the outer
c     boundary of the last cell indexed IN.  The first interface at
c     RADHN(I1) is the outer boundary of the integration domain before
c     the first cell indexed I1.
c
c     Language and Limitations:  LCPFCT is a package of FORTRAN 77 sub-
c     routines written in single precision (64 bits CRAY). The parameter
c     NPT is used to establish the internal FCT array dimensions at the
c     maximum size expected.  Thus NPT = 202 means that continuity equa-
c     tions for systems up to 200 cells long in one direction can be
c     integrated.  Underflows can occur when the function being trans-
c     ported has a region of zeroes.  The calculations misconserve by
c     one or two bits per cycle.  Relative phase and amplitude errors
c     (for smooth functions) are typically a few percent for character-
c     istic lengths of 1 - 2 cells (wavelengths of order 10 cells).  The
c     jump conditions for shocks are generally accurate to better than 1
c     percent.  Common blocks are used to transmit all data between the
c     subroutines in the LCPFCT package.
c
c     Auxiliary Subroutines:  CNVFCT, CONSERVE, COPYGRID, MAKEGRID,
c     NEW_GRID, RESIDIFF, SET_GRID, SOURCES, VELOCITY, ZERODIFF, and
c     ZEROFLUX.  The detailed documentation report provided (or the
c     listing below) explains the definitions and use of the arguments
c     to these other subroutines making up the LCPFCT package.  These
c     routines are not called from LCPFCT itself but are controlled by
c     calls from the user.  Subroutines MAKEGRID, VELOCITY and SOURCES
c     in this package must first be called to set the grid geometry,
```

```
c     velocity-dependent flux and diffusion coefficients, and external
c     source arrays used by LCPFCT.  The other subroutines may be called
c     to perform other functions such as to modify boundary conditions,
c     to perform special grid operations, or compute conservation sums.
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, I1, IN, I1P, INP, I
          Real      BIGNUM, SRHO1, VRHO1, SRHON, VRHON, RHO1M, RHONP
          Real      RHOT1M, RHOTNP, RHOTD1M, RHOTDNP
          Logical   PBC
          Parameter ( NPT = 202 )
          Parameter ( BIGNUM = 1.0E38 )
c     BIGNUM = Machine Dependent Largest Number - Set By The User!!!!

          Real     RHOO(NPT),     RHON(NPT)

c     /FCT_SCRH/ Holds scratch arrays for use by LCPFCT and CNVFCT
          Real     SCRH(NPT),     SCR1(NPT),     DIFF(NPT)
          Real     FLXH(NPT),     FABS(NPT),     FSGN(NPT)
          Real     TERM(NPT),     TERP(NPT),     LNRHOT(NPT)
          Real     LORHOT(NPT),   RHOT(NPT),     RHOTD(NPT)
          Common  /FCT_SCRH/ SCRH, SCR1, DIFF,   FLXH,   FABS, FSGN,
     &                       TERM, TERP, LNRHOT, LORHOT, RHOT, RHOTD

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

c     /FCT_VELO/ Holds velocity-dependent flux coefficients
          Real     HADUDTH(NPT),  NULH(NPT),     MULH(NPT)
          Real     EPSH(NPT),     VDTODR(NPT)
          Common  /FCT_VELO/ HADUDTH, NULH, MULH, EPSH, VDTODR

c     /FCT_MISC/ Holds the source array and diffusion coefficient
          Real     SOURCE(NPT),   DIFF1
          Common  /FCT_MISC/ SOURCE, DIFF1

C-----------------------------------------------------------------------
          I1P = I1 + 1
          INP = IN + 1

c     Calculate the convective and diffusive fluxes . . .
C-----------------------------------------------------------------------
          If ( PBC ) Then
             RHO1M = RHOO(IN)
             RHONP = RHOO(I1)
          Else
             RHO1M = SRHO1*RHOO(I1) + VRHO1
             RHONP = SRHON*RHOO(IN) + VRHON
          End If

          DIFF(I1) = NULH(I1) * ( RHOO(I1) - RHO1M )
          FLXH(I1) = HADUDTH(I1) * ( RHOO(I1) + RHO1M )

          Do 1 I = I1P, IN
             FLXH(I) = HADUDTH(I) * ( RHOO(I) + RHOO(I-1) )
    1        DIFF(I) = NULH(I) * ( RHOO(I) - RHOO(I-1) )
```

```
          DIFF(INP) = NULH(INP) * ( RHONP - RHOO(IN) )
          FLXH(INP) = HADUDTH(INP) * ( RHONP + RHOO(IN) )

c     Calculate LORHOT, the transported mass elements, and LNRHOT, the
c     transported & diffused mass elements  . . .
C-------------------------------------------------------------------------
          Do 2 I = I1, IN
             LORHOT(I) = LO(I)*RHOO(I) + SOURCE(I) + (FLXH(I)-FLXH(I+1))
             LNRHOT(I) = LORHOT(I) + (DIFF(I+1) - DIFF(I))
             RHOT(I)  = LORHOT(I)*RLN(I)
    2        RHOTD(I) = LNRHOT(I)*RLN(I)

c     Evaluate the boundary conditions for RHOT and RHOTD . . .
C-------------------------------------------------------------------------
          If ( PBC ) Then
             RHOT1M = RHOT(IN)
             RHOTNP = RHOT(I1)
             RHOTD1M = RHOTD(IN)
             RHOTDNP = RHOTD(I1)
          Else
             RHOT1M = SRHO1*RHOT(I1) + VRHO1
             RHOTNP = SRHON*RHOT(IN) + VRHON
             RHOTD1M = SRHO1*RHOTD(I1) + VRHO1
             RHOTDNP = SRHON*RHOTD(IN) + VRHON
          End If

c     Calculate the transported antiduffusive fluxes and transported
c     and diffused density differences . . .
C-------------------------------------------------------------------------
          FLXH(I1) = MULH(I1) * ( RHOT(I1) - RHOT1M )
          DIFF(I1) = RHOTD(I1) - RHOTD1M
          FABS(I1) = ABS ( FLXH(I1) )
          FSGN(I1) = SIGN ( DIFF1, DIFF(I1) )

          Do 3 I = I1P, IN
             FLXH(I) = MULH(I) * ( RHOT(I) - RHOT(I-1) )
    3        DIFF(I) = RHOTD(I) - RHOTD(I-1)

          FLXH(INP) = MULH(INP) * ( RHOTNP - RHOT(IN) )
          DIFF(INP) = RHOTDNP - RHOTD(IN)

c     Calculate the magnitude & sign of the antidiffusive flux followed
c     by the flux-limiting changes on the right and left . . .
C-------------------------------------------------------------------------
          Do 4 I = I1, IN
             FABS(I+1) = ABS ( FLXH(I+1) )
             FSGN(I+1) = SIGN ( DIFF1, DIFF(I+1) )
             TERM(I+1) = FSGN(I+1)*LN(I)*DIFF(I)
    4        TERP(I) = FSGN(I)*LN(I)*DIFF(I+1)

          If ( PBC ) Then
             TERP(INP) = TERP(I1)
             TERM(I1) = TERM(INP)
          Else
             TERP(INP) = BIGNUM
             TERM(I1) = BIGNUM
          End If

c     Correct the transported fluxes completely and then calculate the
c     new Flux-Corrected Transport densities . . .
```

```
C-----------------------------------------------------------------------
          FLXH(I1) = FSGN(I1) * AMAX1 ( 0.0,
     &                 AMIN1 ( TERM(I1), FABS(I1), TERP(I1) ) )

          Do 5 I = I1, IN
             FLXH(I+1) = FSGN(I+1) * AMAX1 ( 0.0,
     &                AMIN1 ( TERM(I+1), FABS(I+1), TERP(I+1) ) )
             RHON(I) = RLN(I) * ( LNRHOT(I) + (FLXH(I) - FLXH(I+1)) )
    5        SOURCE(I) = 0.0

       Return
       End

C=======================================================================

       Subroutine MAKEGRID ( RADHO, RADHN, I1, INP, ALPHA )

C-----------------------------------------------------------------------
c
c      Description:  This Subroutine initializes geometry variables and
c      coefficients. It should be called first to initialize the grid.
c      The grid must be defined for all of the grid interfaces from I1 to
c      INP.  Subsequent calls to VELOCITY and LCPFCT can work on only
c      portions of the grid, however, to perform restricted integrations
c      on separate line segments.
c
c      Arguments:
c      RADHO    Real Array(INP)    old cell interface positions         I
c      RADHN    Real Array(INP)    new cell interface positions         I
c      I1       Integer            first cell interface                 I
c      INP      Integer            last cell interface                  I
c      ALPHA    Integer            = 1 for cartesian geometry           I
c                                  = 2 for cylindrical geometry         I
c                                  = 3 for spherical geometry           I
c                                  = 4 general geometry (user supplied) I
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, I1, I1P, I, IN, INP, ALPHA
          Parameter ( NPT = 202 )

          Real      RADHO(INP), RADHN(INP), PI, FTPI

c      /FCT_SCRH/ Holds scratch arrays for use by LCPFCT and CNVFCT
          Real      SCRH(NPT),     SCR1(NPT),     DIFF(NPT)
          Real      FLXH(NPT),     FABS(NPT),     FSGN(NPT)
          Real      TERM(NPT),     TERP(NPT),     LNRHOT(NPT)
          Real      LORHOT(NPT),   RHOT(NPT),     RHOTD(NPT)
          Common  /FCT_SCRH/ SCRH, SCR1, DIFF,   FLXH,   FABS, FSGN,
     &                       TERM, TERP, LNRHOT, LORHOT, RHOT, RHOTD

c      /FCT_GRID/ Holds geometry, grid, area and volume information
          Real      LO(NPT),       LN(NPT),       AH (NPT)
          Real      RLN(NPT),      LH (NPT),      RLH(NPT)
          Real      ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

          DATA      PI, FTPI /3.1415927, 4.1887902/

C-----------------------------------------------------------------------
```

```
          I1P = I1 + 1
          IN = INP - 1

c  Store the old and new grid interface locations from input and then
c  update the new and average interface and grid coefficients . . .
C-------------------------------------------------------------------------
          Do 1 I = I1, INP
             ROH(I) = RADHO(I)
    1        RNH(I) = RADHN(I)

c  Select the choice of coordinate systems . . .
C-------------------------------------------------------------------------
          Go To (100, 200, 300, 400), ALPHA

c  Cartesian coordinates . . .
C-------------------------------------------------------------------------
 100      AH(INP) = 1.0
          Do 101 I = I1, IN
             AH(I) = 1.0
             LO(I) = ROH(I+1) - ROH(I)
  101        LN(I) = RNH(I+1) - RNH(I)
          Go To 500

c  Cylindrical Coordinates: RADIAL . . .
C-------------------------------------------------------------------------
  200     DIFF(I1) = RNH(I1)*RNH(I1)
          SCRH(I1) = ROH(I1)*ROH(I1)
          AH(INP) = PI*(ROH(INP) + RNH(INP))
          DO 201 I = I1, IN
             AH(I) = PI*(ROH(I) + RNH(I))
             SCRH(I+1) = ROH(I+1)*ROH(I+1)
             LO(I) = PI*(SCRH(I+1) - SCRH(I))
             DIFF(I+1) = RNH(I+1)*RNH(I+1)
  201        LN(I) = PI*(DIFF(I+1) - DIFF(I))
          Go To 500

c  Spherical Coordinates: RADIAL . . .
C-------------------------------------------------------------------------
  300     SCR1(I1) = ROH(I1)*ROH(I1)*ROH(I1)
          DIFF(I1) = RNH(I1)*RNH(I1)*RNH(I1)
          SCRH(INP) = (ROH(INP) + RNH(INP))*ROH(INP)
          AH(INP) = FTPI*(SCRH(INP) + RNH(INP)*RNH(INP))
          DO 301 I = I1, IN
             SCR1(I+1) = ROH(I+1)*ROH(I+1)*ROH(I+1)
             DIFF(I+1) = RNH(I+1)*RNH(I+1)*RNH(I+1)
             SCRH(I) = (ROH(I) + RNH(I))*ROH(I)
             AH(I) = FTPI*(SCRH(I) + RNH(I)*RNH(I))
             LO(I) = FTPI*(SCR1(I+1) - SCR1(I))
  301        LN(I) = FTPI*(DIFF(I+1) - DIFF(I))
          Go To 500

c  Special Coordinates: Areas and Volumes are User Supplied . . .
C-------------------------------------------------------------------------
  400     Continue

c  Additional system independent geometric variables . . .
C-------------------------------------------------------------------------
  500     Do 501 I = I1, IN
  501        RLN(I) = 1.0/LN(I)
          LH(I1)  = LN(I1)
          RLH(I1) = RLN(I1)
```

```
          Do 502 I = I1P, IN
             LH(I) =  0.5*(LN(I) + LN(I-1))
  502        RLH(I) = 0.5*(RLN(I) + RLN(I-1))
          LH(INP)  = LN(IN)
          RLH(INP) = RLN(IN)
          Do 503 I = I1, INP
  503        ADUGTH(I) = AH(I)*(RNH(I) - ROH(I))

      Return
      End

C=======================================================================

      Subroutine VELOCITY ( UH, I1, INP, DT )

C-----------------------------------------------------------------------
c
c     Description:   This subroutine calculates all velocity-dependent
c     coefficients for the LCPFCT and CNVFCT routines. This routine
c     must be called before either LCPFCT or CNVFCT is called.  MAKEGRID
c     must be called earlier to set grid and geometry data used here.
c
c     Arguments:
c     UH     Real Array(NPT)   flow velocity at cell interfaces        I
c     I1     Integer           first cell interface of integration     I
c     INP    Integer           last cell interface = N + 1             I
c     DT     Real              stepsize for the time integration       I
c
C-----------------------------------------------------------------------

          Implicit NONE
          Integer  NPT, I1, I1P, I, IN, INP
          Parameter ( NPT = 202 )

          Real     UH(INP), DT, RDT, DTH, DT2, DT4, ONE3RD, ONE6TH

c     /FCT_SCRH/ Holds scratch arrays for use by LCPFCT and CNVFCT
          Real     SCRH(NPT),     SCR1(NPT),     DIFF(NPT)
          Real     FLXH(NPT),     FABS(NPT),     FSGN(NPT)
          Real     TERM(NPT),     TERP(NPT),     LNRHOT(NPT)
          Real     LORHOT(NPT),   RHOT(NPT),     RHOTD(NPT)
          Common  /FCT_SCRH/ SCRH, SCR1, DIFF,   FLXH,   FABS, FSGN,
     &                       TERM, TERP, LNRHOT, LORHOT, RHOT, RHOTD

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

c     /FCT_VELO/ Holds velocity-dependent flux coefficients
          Real     HADUDTH(NPT),  NULH(NPT),     MULH(NPT)
          Real     EPSH(NPT),     VDTODR(NPT)
          Common  /FCT_VELO/ HADUDTH, NULH, MULH, EPSH, VDTODR

C-----------------------------------------------------------------------
          I1P = I1 + 1
          IN = INP - 1

c     Calculate 0.5*Interface Area * Velocity Difference * DT (HADUDTH).
c     Next calculate the interface epsilon (EPSH = V*DT/DX).  Then find
```

```
c     the diffusion (NULH) and antidiffusion (MULH) coefficients.  The
c     variation with epsilon gives fourth-order accurate phases when the
c     grid is uniform, the velocity constant, and SCRH is set to zero.
c     With SCRH nonzero (as below) slightly better results are obtained
c     in some of the tests.  Optimal performance, of course, depends on
c     on the application.
C-----------------------------------------------------------------------
          RDT = 1.0/DT
          DTH = 0.5*DT
          ONE6TH = 1.0/6.0
          ONE3RD = 1.0/3.0
          Do 1 I = I1, INP
             HADUDTH(I) = DT*AH(I)*UH(I) - ADUGTH(I)
             EPSH(I) = HADUDTH(I)*RLH(I)
             SCRH(I) = AMIN1 ( ONE6TH, ABS(EPSH(I)) )
             SCRH(I) = ONE3RD*SCRH(I)**2
             HADUDTH(I) = 0.5*HADUDTH(I)
             NULH(I) =  ONE6TH + ONE3RD*(EPSH(I) + SCRH(I))*
     &                                  (EPSH(I) - SCRH(I))
             MULH(I) =  0.25 - 0.5*NULH(I)
             NULH(I) = LH(I)*(NULH(I) + SCRH(I))
             MULH(I) = LH(I)*(MULH(I) + SCRH(I))
    1        DIFF(I) = UH(I) - RDT*(RNH(I) - ROH(I))

c     Now calculate VDTODR for CNVFCT . . .
C-----------------------------------------------------------------------
          DT2 = 2.0*DT
          DT4 = 4.0*DT
          VDTODR(I1) = DT2*DIFF(I1)/(RNH(I1P)-RNH(I1) +
     &                               ROH(I1P)-ROH(I1))

          Do 2 I = I1P, IN
    2        VDTODR(I) = DT4*DIFF(I)/(RNH(I+1)-RNH(I-1) +
     &                                ROH(I+1)-ROH(I-1))

          VDTODR(INP) = DT2*DIFF(INP)/(RNH(INP)-RNH(IN) +
     &                                 ROH(INP)-ROH(IN))

        Return
        End

C=======================================================================

      Subroutine SOURCES ( I1, IN, DT, MODE, C, D, D1, DN)

C-----------------------------------------------------------------------
c
c     Description:   This Subroutine accumulates different source terms.
c
c     Arguments:
c     I1     Integer          first cell to be integrated            I
c     IN     Integer          last cell  to be integrated            I
c     DT     Real             stepsize for the time integration      I
c     MODE   Integer          = 1   computes + DIV (D)               I
c                             = 2   computes + C*GRAD (D)            I
c                             = 3   adds + D to the sources          I
c                             = 4   + DIV (D) from interface data    I
c                             = 5   + C*GRAD (D) from interface data I
c                             = 6   + C for list of scalar indices   I
c     C      Real Array(NPT)  Array of source variables              I
c     D      Real Array(NPT)  Array of source variables              I
```

```
c     D1     Real              first boundary value of D              I
c     DN     Real              last  boundary value of D              I
c
C-----------------------------------------------------------------------

          Implicit NONE
          Integer  NPT, NINDMAX, MODE, IS, I, I1, IN, I1P, INP
          Parameter ( NPT = 202, NINDMAX = 150 )

          Real     C(NPT), D(NPT), DT, DTH, DTQ, D1, DN

c     /FCT_NDEX/ Holds a scalar list of special cell information . . .
          Real     SCALARS(NINDMAX)
          Integer  INDEX(NINDMAX), NIND
          Common  /FCT_NDEX/ NIND, INDEX, SCALARS

c     /FCT_SCRH/ Holds scratch arrays for use by LCPFCT and CNVFCT
          Real     SCRH(NPT),     SCR1(NPT),     DIFF(NPT)
          Real     FLXH(NPT),     FABS(NPT),     FSGN(NPT)
          Real     TERM(NPT),     TERP(NPT),     LNRHOT(NPT)
          Real     LORHOT(NPT),   RHOT(NPT),     RHOTD(NPT)
          Common  /FCT_SCRH/ SCRH, SCR1, DIFF,   FLXH,   FABS, FSGN,
     &                       TERM, TERP, LNRHOT, LORHOT, RHOT, RHOTD

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

c     /FCT_MISC/ Holds the source array and diffusion coefficient
          Real     SOURCE(NPT),   DIFF1
          Common  /FCT_MISC/ SOURCE, DIFF1

C-----------------------------------------------------------------------
          I1P = I1 + 1
          INP = IN + 1
          DTH = 0.5*DT
          DTQ = 0.25*DT
          Go To  ( 101, 202, 303, 404, 505, 606 ), MODE

c  + DIV(D) is computed conservatively and added to SOURCE . . .
C-----------------------------------------------------------------------
  101     SCRH(I1) = DT*AH(I1)*D1
          SCRH(INP) = DT*AH(INP)*DN
          Do 1 I = IN, I1P, -1
             SCRH(I) = DTH*AH(I)*(D(I) + D(I-1))
    1        SOURCE(I) = SOURCE(I) + (SCRH(I+1) - SCRH(I))
          SOURCE(I1) = SOURCE(I1) + (SCRH(I1P) - SCRH(I1))
       Return

c  + C*GRAD(D) is computed efficiently and added to the SOURCE . . .
C-----------------------------------------------------------------------
  202     SCRH(I1) = DTH*D1
          SCRH(INP) = DTH*DN
          Do 2 I = IN, I1P, -1
             SCRH(I) = DTQ*(D(I)+D(I-1))
             DIFF(I) = SCRH(I+1) - SCRH(I)
    2        SOURCE(I) = SOURCE(I)
     &                  + C(I)*(AH(I+1)+AH(I))*DIFF(I)
          SOURCE(I1) = SOURCE(I1) + C(I1)*(AH(I1P)+AH(I1))*
```

```
     &                  (SCRH(I1P)-SCRH(I1))
      Return

c  + D is added to SOURCE in an explicit formulation . . .
C-----------------------------------------------------------------------
  303    Do 3 I = I1, IN
    3       SOURCE(I) = SOURCE(I) + DT*LO(I)*D(I)
      Return

c  + DIV(D) is computed conservatively from interface data . . .
C-----------------------------------------------------------------------
  404    SCRH(INP) = DT*AH(INP)*DN
         SCRH( I1) = DT*AH( I1)*D1
         Do 4 I = IN, I1P, -1
            SCRH(I)   = DT*AH(I)*D(I)
    4       SOURCE(I) = SOURCE(I)+SCRH(I+1)-SCRH(I)
         SOURCE(I1) = SOURCE(I1) + SCRH(I1P) - SCRH(I1)
      Return

c  + C*GRAD(D) is computed using interface data . . .
C-----------------------------------------------------------------------
  505    SCRH( I1) = DTH*D1
         SCRH(INP) = DTH*DN
         Do 5 I = IN, I1P, -1
            SCRH(I) = DTH*D(I)
            DIFF(I) = SCRH(I+1) - SCRH(I)
    5       SOURCE(I) = SOURCE(I)
     &                  + C(I)*(AH(I+1)+AH(I))*DIFF(I)
         SOURCE(I1) = SOURCE(I1) + C(I1)*(AH(I1P)+AH(I1))*
     &                       (SCRH(I1P)-SCRH(I1))
      Return

c  + C for source terms only at a list of indices . . .
C-----------------------------------------------------------------------
  606    Do 6 IS = 1, NIND
            I = INDEX(IS)
    6       SOURCE(I) = SOURCE(I) + SCALARS(IS)

      Return
      End

C=======================================================================

      Subroutine CNVFCT ( RHOO, RHON, I1, IN,
     &                    SRHO1, VRHO1, SRHON, VRHON, PBC )

C-----------------------------------------------------------------------
c
c     Originated: J.P. Boris         Code 4400, NRL          Feb 1987
c     Modified:  Laboratory for Computational Physics & Fluid Dynamics
c     Contact:    J.P. Boris, J.H. Gardner, A.M. Landsberg, or E.S. Oran
c
c     Description:  This routine solves an advective continuity equation
c     of the form  dRHO/dt = -V*grad(RHO) + SOURCES in the user's choice
c     of Cartesian, cylindrical, or spherical coordinate systems.  A
c     facility is included to allow definition of other coordinates.
c     The grid can be Eulerian, sliding rezone, or Lagrangian and can
c     be arbitrarily spaced.  The algorithm is a low-phase-error FCT
c     algorithm, vectorized and optimized for a combination of speed and
c     flexibility.  A complete description appears in the NRL Memorandum
c     Report (1992), "LCPFCT - A Flux-Corrected Transport Algorithm For
```

```
c     Solving Generalized Continuity Equations".
c
c     Arguments:
c     RHOO   Real Array        grid point densities at start of step   I
c     RHON   Real Array        grid point densities at end of step     O
c     I1     Integer           first grid point of integration         I
c     IN     Integer           last grid point of intergration         I
c     SRHO1  Real Array        boundary guard cell factor at cell I1+1 I
c     VRHO1  Real Array        boundary value added to guard cell I1-1 I
c     SRHON  Real Array        boundary guard cell factor at cell IN+1 I
c     VRHON  Real Array        boundary value added to guard cell IN+1 I
c     PBC    Logical           periodic boundaries if PBC = .true.     I
c
c        In this routine the last interface at RADHN(INP) is the outer
c     boundary of the last cell indexed IN.  The first interface at
c     RADHN(I1) is the outer boundary of the integration domain before
c     the first cell indexed I1.  The description of CNVFCT and the
c     roles played by the auxiliary library routines is the same for
c     LCPFCT given above.
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, I1, IN, I1P, INP, I
          Real      BIGNUM, SRHO1, VRHO1, SRHON, VRHON, RHO1M, RHONP
          Real      RHOT1M, RHOTNP, RHOTD1M, RHOTDNP
          Logical   PBC
          Parameter ( NPT = 202 )
          Parameter ( BIGNUM = 1.0E38 )
c     BIGNUM = Machine Dependent Largest Number - Set By The User!!!!

          Real     RHOO(NPT),     RHON(NPT)

c     /FCT_SCRH/ Holds scratch arrays for use by LCPFCT and CNVFCT
          Real     SCRH(NPT),     SCR1(NPT),     DIFF(NPT)
          Real     FLXH(NPT),     FABS(NPT),     FSGN(NPT)
          Real     TERM(NPT),     TERP(NPT),     LNRHOT(NPT)
          Real     LORHOT(NPT),   RHOT(NPT),     RHOTD(NPT)
          Common  /FCT_SCRH/ SCRH, SCR1, DIFF,   FLXH,   FABS, FSGN,
     &                       TERM, TERP, LNRHOT, LORHOT, RHOT, RHOTD

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

c     /FCT_VELO/ Holds velocity-dependent flux coefficients
          Real     HADUDTH(NPT),  NULH(NPT),     MULH(NPT)
          Real     EPSH(NPT),     VDTODR(NPT)
          Common  /FCT_VELO/ HADUDTH, NULH, MULH, EPSH, VDTODR

c     /FCT_MISC/ Holds the source array and diffusion coefficient
          Real     SOURCE(NPT),   DIFF1
          Common  /FCT_MISC/ SOURCE, DIFF1

C-----------------------------------------------------------------------
          I1P = I1 + 1
          INP = IN + 1

c     Calculate the convective and diffusive fluxes . . .
```

```
C-----------------------------------------------------------------------
          If ( PBC ) Then
             RHO1M = RHOO(IN)
             RHONP = RHOO(I1)
          Else
             RHO1M = SRHO1*RHOO(I1) + VRHO1
             RHONP = SRHON*RHOO(IN) + VRHON
          End If

          DIFF(I1) = NULH(I1) * ( RHOO(I1) - RHO1M )
          FLXH(I1) = VDTODR(I1) * ( RHOO(I1) - RHO1M )

          Do 1 I = I1P, IN
             DIFF(I) = ( RHOO(I) - RHOO(I-1) )
             FLXH(I) = VDTODR(I) * DIFF(I)
    1        DIFF(I) = NULH(I) * DIFF(I)

          DIFF(INP) = NULH(INP) * ( RHONP - RHOO(IN) )
          FLXH(INP) = VDTODR(INP) * ( RHONP - RHOO(IN) )

c     Calculate LORHOT, the transported mass elements, and LNRHOT, the
c     transported & diffused mass elements  . . .
C-----------------------------------------------------------------------
          Do 2 I = I1, IN
             LORHOT(I) = LN(I) * (RHOO(I) - 0.5*(FLXH(I+1) + FLXH(I)))
     &                    + SOURCE(I)
             LNRHOT(I) = LORHOT(I) + (DIFF(I+1) - DIFF(I))
             RHOT(I)  = LORHOT(I)*RLN(I)
    2        RHOTD(I) = LNRHOT(I)*RLN(I)

c     Evaluate the boundary conditions for RHOT and RHOTD . . .
C-----------------------------------------------------------------------
          If ( PBC ) Then
             RHOT1M = RHOT(IN)
             RHOTNP = RHOT(I1)
             RHOTD1M = RHOTD(IN)
             RHOTDNP = RHOTD(I1)
          Else
             RHOT1M = SRHO1*RHOT(I1) + VRHO1
             RHOTNP = SRHON*RHOT(IN) + VRHON
             RHOTD1M = SRHO1*RHOTD(I1) + VRHO1
             RHOTDNP = SRHON*RHOTD(IN) + VRHON
          End If

c     Calculate the transported antiduffusive fluxes and transported
c     and diffused density differences . . .
C-----------------------------------------------------------------------
          FLXH(I1) = MULH(I1) * ( RHOT(I1) - RHOT1M )
          DIFF(I1) = RHOTD(I1) - RHOTD1M
          FABS(I1) = ABS ( FLXH(I1) )
          FSGN(I1) = SIGN ( DIFF1, DIFF(I1) )

          Do 3 I = I1P, IN
             FLXH(I) = MULH(I) * ( RHOT(I) - RHOT(I-1) )
    3        DIFF(I) = RHOTD(I) - RHOTD(I-1)

          FLXH(INP) = MULH(INP) * ( RHOTNP - RHOT(IN) )
          DIFF(INP) = RHOTDNP - RHOTD(IN)

c     Calculate the magnitude & sign of the antidiffusive flux followed
c     by the flux-limiting changes on the right and left . . .
```

```
C-----------------------------------------------------------------------
          Do 4 I = I1, IN
             FABS(I+1) = ABS ( FLXH(I+1) )
             FSGN(I+1) = SIGN ( DIFF1, DIFF(I+1) )
             TERM(I+1) = FSGN(I+1)*LN(I)*DIFF(I)
    4        TERP(I) = FSGN(I)*LN(I)*DIFF(I+1)

          If ( PBC ) Then
             TERP(INP) = TERP(I1)
             TERM(I1) = TERM(INP)
          Else
             TERP(INP) = BIGNUM
             TERM(I1) = BIGNUM
          End If

c     Correct the transported fluxes completely and then calculate the
c     new Flux-Corrected Transport densities . . .
C-----------------------------------------------------------------------
          FLXH(I1) = FSGN(I1) * AMAX1 ( 0.0,
     &                  AMIN1 ( TERM(I1), FABS(I1), TERP(I1) ) )

          Do 5 I = I1, IN
             FLXH(I+1) = FSGN(I+1) * AMAX1 ( 0.0,
     &                 AMIN1 ( TERM(I+1), FABS(I+1), TERP(I+1) ) )
             RHON(I) = RLN(I) * ( LNRHOT(I) + (FLXH(I) - FLXH(I+1)) )
    5        SOURCE(I) = 0.0

       Return
       End

C=======================================================================

       Subroutine CONSERVE ( RHO, I1, IN, CSUM )

C-----------------------------------------------------------------------
c
c     Description:   This routine computes the ostensibly conserved sum.
c     Beware your boundary conditions and note that only one continuity
c     equation is summed for each call to this subroutine.
c
c     Arguments:
c     RHO    Real Array(NPT)   cell values for physical variable 'RHO' I
c     I1     Integer           first cell to be integrated             I
c     IN     Integer           last cell  to be integrated             I
c     CSUM   Real              value of the conservation sum of rho    O
c
C-----------------------------------------------------------------------

          Implicit NONE
          Integer  NPT, I, I1, IN
          Parameter ( NPT = 202 )
          Real     CSUM, RHO(NPT)

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

c  Compute the ostensibly conserved total mass (BEWARE B.C.) . . .
C-----------------------------------------------------------------------
```

```
          CSUM = 0.0
          Do 80 I = I1, IN
 80          CSUM = CSUM + LN(I)*RHO(I)

      Return
      End

C=======================================================================

      Subroutine COPYGRID ( MODE, I1, IN )

C-----------------------------------------------------------------------
c
c     Description:   This Subroutine makes a complete copy of the grid
c     variables defined by the most recent call to MAKEGRID from cell
c     I1 to IN including the boundary values at interface IN+1 when the
c     argument MODE = 1.  When MODE = 2, these grid variables are reset
c     from common block OLD_GRID.  This routine is used where the same
c     grid is needed repeatedly after some of the values have been over-
c     written, for example, by a grid which moves between the halfstep
c     and the whole step.
c
c     Argument:
c     I1       Integer         first cell index                        I
c     IN       Integer         last cell index                         I
c     MODE     Integer         = 1 grid variables copied into OLD_GRID I
c                              = 2 grid restored from OLD_GRID common  I
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, I, MODE, I1, IN
          Parameter ( NPT = 202 )

c     /OLD_GRID/ Holds geometry, grid, area and volume information
          Real     LOP(NPT),      LNP(NPT),      AHP(NPT)
          Real     RLNP(NPT),     RLHP(NPT),     LHP(NPT)
          Real     ROHP(NPT),     RNHP(NPT),     ADUGTHP(NPT)
          Common  /OLD_GRID/ LOP, LNP, AHP, RLNP, LHP, RLHP,
     &                       ROHP,     RNHP,      ADUGTHP

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

C-----------------------------------------------------------------------
          If ( MODE .eq. 1 ) Then
             Do 101 I = I1, IN
                LOP(I)  = LO(I)
                LNP(I)  = LN(I)
  101           RLNP(I) = RLN(I)
             Do 102 I = I1, IN+1
                AHP(I)  = AH(I)
                LHP(I)  = LH(I)
                RLHP(I) = RLH(I)
                ROHP(I) = ROH(I)
                RNHP(I) = RNH(I)
  102           ADUGTHP(I) = ADUGTH(I)
          Else If ( MODE .eq. 2 ) Then
```

```
               Do 201 I = I1, IN
                  LO(I)  = LOP(I)
                  LN(I)  = LNP(I)
  201             RLN(I) = RLNP(I)
               Do 202 I = I1, IN+1
                  AH(I)  = AHP(I)
                  LH(I)  = LHP(I)
                  RLH(I) = RLHP(I)
                  ROH(I) = ROHP(I)
                  RNH(I) = RNHP(I)
  202             ADUGTH(I) = ADUGTHP(I)
            Else
               Write ( 6, 1001 ) MODE
            End If
 1001       Format ( ' COPYGRID Error! MODE =', I3, ' (not 1 or 2!)' )

      Return
      End

C=======================================================================

      Block Data FCTBLK

C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT
          Parameter ( NPT = 202 )

c     /FCT_MISC/ Holds the source array and diffusion coefficient
          Real     SOURCE(NPT),   DIFF1
          Common  /FCT_MISC/ SOURCE, DIFF1

          Data SOURCE / NPT*0.0 /, DIFF1 / 0.999 /

      End

C=======================================================================

      Subroutine NEW_GRID ( RADHN, I1, INP, ALPHA )

C-----------------------------------------------------------------------
c
c     Description:  This Subroutine initializes geometry variables and
c     coefficients when the most recent call to MAKEGRID used the same
c     set of values RADHO and only the new interface locations RADHN are
c     different.  NEW_GRID is computationally more efficienty than the
c     complete grid procedure in MAKEGRID because several formulae do
c     not need to be recomputed.  The grid should generally be defined
c     for the entire number of grid interfaces from 1 to INP,  however
c     subsets of the entire grid may be reinitialized with care.
c
c     Arguments:
c     RADHN    Real Array(INP)    new cell interface positions         I
c     I1       Integer            first interface index                I
c     INP      Integer            last interface index                 I
c     ALPHA    Integer            = 1 for cartesian geometry           I
c                                 = 2 for cylindrical geometry         I
c                                 = 3 for spherical geometry           I
c                                 = 4 general geometry (user supplied) I
c
```

```
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, I1, I1P, I, IN, INP, ALPHA
          Parameter ( NPT = 202 )

          Real     RADHN(INP), PI, FTPI

c     /FCT_SCRH/ Holds scratch arrays for use by LCPFCT and CNVFCT
          Real     SCRH(NPT),     SCR1(NPT),     DIFF(NPT)
          Real     FLXH(NPT),     FABS(NPT),     FSGN(NPT)
          Real     TERM(NPT),     TERP(NPT),     LNRHOT(NPT)
          Real     LORHOT(NPT),   RHOT(NPT),     RHOTD(NPT)
          Common  /FCT_SCRH/ SCRH, SCR1, DIFF,   FLXH,   FABS, FSGN,
     &                       TERM, TERP, LNRHOT, LORHOT, RHOT, RHOTD

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

          DATA     PI, FTPI /3.1415927, 4.1887902/

C-----------------------------------------------------------------------
          I1P = I1 + 1
          IN = INP - 1

c  Store the old and new grid interface locations from input and then
c  update the new and average interface and grid coefficients . . .
C-----------------------------------------------------------------------
          Do 1 I = I1, INP
    1        RNH(I) = RADHN(I)

c  Select the choice of coordinate systems . . .
C-----------------------------------------------------------------------
          Go To (100, 200, 300, 400), ALPHA

c  Cartesian coordinates . . .
C-----------------------------------------------------------------------
 100      AH(INP) = 1.0
          Do 101 I = I1, IN
  101        LN(I) = RNH(I+1) - RNH(I)
          Go To 500

c  Cylindrical Coordinates: RADIAL . . .
C-----------------------------------------------------------------------
  200     DIFF(I1) = RNH(I1)*RNH(I1)
          AH(INP) = PI*(ROH(INP) + RNH(INP))
          DO 201 I = I1, IN
             AH(I) = PI*(ROH(I) + RNH(I))
             DIFF(I+1) = RNH(I+1)*RNH(I+1)
  201        LN(I) = PI*(DIFF(I+1) - DIFF(I))
          Go To 500

c  Spherical Coordinates: RADIAL . . .
C-----------------------------------------------------------------------
  300     DIFF(I1) = RNH(I1)*RNH(I1)*RNH(I1)
          SCRH(INP) = (ROH(INP) + RNH(INP))*ROH(INP)
          AH(INP) = FTPI*(SCRH(INP) + RNH(INP)*RNH(INP))
          DO 301 I = I1, IN
```

```
               DIFF(I+1) = RNH(I+1)*RNH(I+1)*RNH(I+1)
               SCRH(I) = (ROH(I) + RNH(I))*ROH(I)
               AH(I) = FTPI*(SCRH(I) + RNH(I)*RNH(I))
  301          LN(I) = FTPI*(DIFF(I+1) - DIFF(I))
           Go To 500

c  Special Coordinates: Areas and Volumes are User Supplied . . .
C-------------------------------------------------------------------------
  400     Continue

c  Additional system independent geometric variables . . .
C-------------------------------------------------------------------------
  500     Do 501 I = I1, IN
  501        RLN(I) = 1.0/LN(I)
           LH(I1)  = LN(I1)
           RLH(I1) = RLN(I1)
           Do 502 I = I1P, IN
              LH(I) =  0.5*(LN(I) + LN(I-1))
  502        RLH(I) = 0.5*(RLN(I) + RLN(I-1))
           LH(INP)  = LN(IN)
           RLH(INP) = RLN(IN)
           Do 503 I = I1, INP
  503        ADUGTH(I) = AH(I)*(RNH(I) - ROH(I))

       Return
       End

C=========================================================================

       Subroutine RESIDIFF ( DIFFA )

C-------------------------------------------------------------------------
c
c      Description:   Allows the user to give FCT some residual numerical
c      diffusion by making the anti-diffusion coefficient smaller.
c
c      Arguments:
c      DIFFA  Real    Replacement residual diffusion coefficient        I
c                     Defaults to 0.999 but could be as high as 1.0000
c
C-------------------------------------------------------------------------

           Implicit NONE
           Integer  NPT
           Real     DIFFA
           Parameter ( NPT = 202 )

c      /FCT_MISC/ Holds the source array and diffusion coefficient
           Real     SOURCE(NPT),   DIFF1
           Common  /FCT_MISC/ SOURCE, DIFF1

           DIFF1 = DIFFA

       Return
       End

C=========================================================================

       Subroutine SET_GRID ( RADR, I1, IN )

C-------------------------------------------------------------------------
```

```
c
c     Description:  This subroutine includes the radial factor in the
c     cell volume for polar coordinates. It must be preceeded by a call
c     to MAKE_GRID with ALPHA = 1 to establish the angular dependence of
c     the cell volumes and areas and a call to COPY_GRID to save this
c     angular dependence.  The angular coordinate is measured in radians
c     (0 to 2 pi) in cylindrical coordinates and cos theta (1 to -1) in
c     spherical coordinates.  SET_GRID is called inside the loop over
c     radius in a multidimensional model to append the appropriate
c     radial factors when integrating in the angular direction.
c
c     Arguments:
c     RADR     Real               radius of cell center               I
c     I1       Integer            first cell index                    I
c     IN       Integer            last cell index                     I
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, I1, I1P, I, IN, INP
          Real      RADR
          Parameter ( NPT = 202 )

c     /OLD_GRID/ Holds geometry, grid, area and volume information
          Real     LOP(NPT),      LNP(NPT),      AHP(NPT)
          Real     RLNP(NPT),     RLHP(NPT),     LHP(NPT)
          Real     ROHP(NPT),     RNHP(NPT),     ADUGTHP(NPT)
          Common  /OLD_GRID/ LOP, LNP, AHP, RLNP, LHP, RLHP,
     &                       ROHP,     RNHP,      ADUGTHP

c     /FCT_GRID/ Holds geometry, grid, area and volume information
          Real     LO(NPT),       LN(NPT),       AH (NPT)
          Real     RLN(NPT),      LH (NPT),      RLH(NPT)
          Real     ROH(NPT),      RNH(NPT),      ADUGTH(NPT)
          Common  /FCT_GRID/ LO, LN, AH, RLN, LH, RLH, ROH, RNH, ADUGTH

C-----------------------------------------------------------------------
          I1P = I1 + 1
          INP = IN + 1

C  Multiply each volume element by the local radius
          DO 100 I = I1, IN
          LN(I) = LNP(I)*RADR
  100     LO(I) = LOP(I)*RADR

c  Additional system independent geometric variables . . .
C-----------------------------------------------------------------------
  500     Do 501 I = I1, IN
  501        RLN(I) = 1.0/LN(I)
          LH(I1)  = LN(I1)
          RLH(I1) = RLN(I1)
          Do 502 I = I1P, IN
             LH(I) =  0.5*(LN(I) + LN(I-1))
  502        RLH(I) = 0.5*(RLN(I) + RLN(I-1))
          LH(INP)  = LN(IN)
          RLH(INP) = RLN(IN)

      Return
      End

C=======================================================================
```

```
      Subroutine ZERODIFF ( IND )

C-----------------------------------------------------------------------
c
c     Description:   This Subroutine sets the FCT diffusion and anti-
c     diffusion parameters to zero at the specified cell interface to
c     inhibit unwanted diffusion across the interface.  This routine is
c     used for inflow and outflow boundary conditions.  If argument IND
c     is positive, the coefficients at that particular interface are
c     reset.  If IND is negative, the list of NIND indices in INDEX are
c     used to reset that many interface coefficients.
c
c     Argument:
c     IND    Integer              index of interface to be reset       I
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, NINDMAX, IND, IS, I
          Parameter ( NPT = 202, NINDMAX = 150 )

c     /FCT_NDEX/ Holds a scalar list of special cell information . . .
          Real     SCALARS(NINDMAX)
          Integer  INDEX(NINDMAX), NIND
          Common  /FCT_NDEX/ NIND, INDEX, SCALARS

c     /FCT_VELO/ Holds velocity-dependent flux coefficients
          Real     HADUDTH(NPT),  NULH(NPT),     MULH(NPT)
          Real     EPSH(NPT),     VDTODR(NPT)
          Common  /FCT_VELO/ HADUDTH, NULH, MULH, EPSH, VDTODR

C-----------------------------------------------------------------------
          If ( IND .gt. 0 ) Then
              NULH(IND) = 0.0
              MULH(IND) = 0.0
          Else If ( IND .le. 0 ) Then
             If ( NIND.lt.1 .or. NIND.gt.NINDMAX .or. IND.eq.0 ) Then
                Write ( 6,* ) ' ZERODIFF Error! IND, NIND =', IND, NIND
                Stop
             End If
             Do IS = 1, NIND
                I = INDEX(IS)
                NULH(I) = 0.0
                MULH(I) = 0.0
             End Do
          End If

      Return
      End

C=======================================================================

      Subroutine ZEROFLUX ( IND )

C-----------------------------------------------------------------------
c
c     Description:   This Subroutine sets all the velocity dependent FCT
c     parameters to zero at the specified cell interface to inhibit
c     transport fluxes AND diffusion of material across the interface.
c     This routine is needed in solid wall boundary conditions.  If IND
```

```
c     is positive, the coefficients at that particular interface are
c     reset.  If IND is negative, the list of NIND indices in INDEX are
c     used to reset that many interface coefficients.
c
c     Argument:
c     IND    Integer              index of interface to be reset       I
c
C-----------------------------------------------------------------------

          Implicit  NONE
          Integer   NPT, NINDMAX, IND, IS, I
          Parameter ( NPT = 202, NINDMAX = 150 )

c     /FCT_NDEX/ Holds a scalar list of special cell information . . .
          Real     SCALARS(NINDMAX)
          Integer  INDEX(NINDMAX), NIND
          Common  /FCT_NDEX/ NIND, INDEX, SCALARS

c     /FCT_VELO/ Holds velocity-dependent flux coefficients
          Real     HADUDTH(NPT),  NULH(NPT),     MULH(NPT)
          Real     EPSH(NPT),     VDTODR(NPT)
          Common  /FCT_VELO/ HADUDTH, NULH, MULH, EPSH, VDTODR

C-----------------------------------------------------------------------
          If ( IND .gt. 0 ) Then
             HADUDTH(IND) = 0.0
             NULH(IND) = 0.0
             MULH(IND) = 0.0
          Else If ( IND .le. 0 ) Then
             If ( NIND.lt.1 .or. NIND.gt.NINDMAX .or. IND.eq.0 ) Then
                Write ( 6,* ) ' ZEROFLUX Error! IND, NIND =', IND, NIND
                Stop
             End If
             Do IS = 1, NIND
                I = INDEX(IS)
                HADUDTH(I) = 0.0
                NULH(I) = 0.0
                MULH(I) = 0.0
             End Do
          End If

      Return
      End

C=======================================================================
```

```
C=======================================================================

      Program CONVECT

C-----------------------------------------------------------------------
c
c  CONSTANT VELOCITY CONVECTION - LCPFCT Test # 1            August 1992
c
c  This program runs three periodic convection problems using LCPFCT and
c  the FCT utility routines .  The three profiles are the square wave, a
c  semicirle, and a Gaussian peak. The velocity is constant in space and
c  time and the grid is kept stationary.
c
C-----------------------------------------------------------------------

         Implicit   NONE

         Integer    NPT,           NX,            NXP
         Parameter  ( NPT = 202 )
         Logical    USE_LCP
         Integer    ISTEP,         JSTEP,         I
         Integer    MAXSTP,        IPRINT,        LOUT
         Real       DX, DT,        VELX,          TIME, XCELL
         Real       CSQUARE,       CCIRCLE,       CGAUSSP
         Real       ISQUARE,       ICIRCLE,       IGAUSSP
         Real       ESQUARE,       ECIRCLE,       EGAUSSP
         Real       ASQUARE(NPT),  ACIRCLE(NPT),  AGAUSSP(NPT)
         Real       SQUARE(NPT),   CIRCLE(NPT),   GAUSSP(NPT)
         Real       XINT(NPT),     VINT(NPT)

 1000 Format ( '1',/,'    LCPFCT Test #1 - Constant V Convection:',
     &         '   step =', I4, ' and TIME =', F7.3, /, 10X,
     &         'with DX =', F6.3, ' DT =', F6.3, ' and VX =', F6.3, /)
 1001 Format ( '    I     X(I)     Square    exact     Circle ',
     &         '   exact   Gaussian    exact' )
 1002 Format ( I5, 7F10.5 )
 1003 Format ( 1X, /, ' Conserved sums', 6F10.5 )
 1004 Format (        ' Absolute error', F10.5, 10X, F10.5, 10X, F10.5 )

c  The Constant Velocity Convection control parameters are initialized.
c  (change here for other cases) . . .
C-----------------------------------------------------------------------
      USE_LCP = .true. ! Use the LCPFCT routine rather than CNVFCT
      NX      =   50   ! Number of cells in the periodic system
      DX      =  1.0   ! Cell size
      DT      =  0.2   ! Timestep for the calculation
      VELX    =  1.0   ! Constant X velocity, VELX*DT/DX = 0.2
      MAXSTP  =  501   ! Number of timesteps, two cycles of the system
      LOUT    =   11   ! Logical unit number of printed output device
      IPRINT  =  125   ! Printout frequency, fluid moves 25 cells

c  The grid, velocity, and three density profiles are initialized . . .
C-----------------------------------------------------------------------
      NXP = NX + 1
      Do 1 I = 1, NXP
         XINT(I) = FLOAT(I-1)*DX
    1    VINT(I) = VELX

      Call PROFILE ( 1, TIME, SQUARE, XINT, NX, NXP, VELX )
      Call PROFILE ( 2, TIME, CIRCLE, XINT, NX, NXP, VELX )
      Call PROFILE ( 3, TIME, GAUSSP, XINT, NX, NXP, VELX )
```

```
c  Set residual diffusion, grid, and velocity factors in LCPFCT . . .
C-----------------------------------------------------------------------
      Call RESIDIFF ( 1.0000 )
      Call MAKEGRID ( XINT, XINT, 1, NXP, 1 )
      Call VELOCITY ( VINT, 1, NXP, DT )

c  Begin loop over timesteps . . .
C-----------------------------------------------------------------------
      TIME = 0.0
      Do 9999 ISTEP = 1, MAXSTP

c  Results are printed as required . . .
C-----------------------------------------------------------------------
         If ( MOD(ISTEP-1,IPRINT) .eq. 0 ) Then
            JSTEP = ISTEP - 1
            Write ( LOUT, 1000 ) JSTEP, TIME, DX, DT, VELX
            Write ( LOUT, 1001 )
            Call PROFILE ( 1, TIME, ASQUARE, XINT, NX,NXP, VELX )
            Call PROFILE ( 2, TIME, ACIRCLE, XINT, NX,NXP, VELX )
            Call PROFILE ( 3, TIME, AGAUSSP, XINT, NX,NXP, VELX )
            ESQUARE = 0.0
            ECIRCLE = 0.0
            EGAUSSP = 0.0
            Do 100 I = 1, NX
               XCELL = XINT(I) + 0.5*DX
               ESQUARE = ESQUARE + ABS(SQUARE(I) - ASQUARE(I))
               ECIRCLE = ECIRCLE + ABS(CIRCLE(I) - ACIRCLE(I))
               EGAUSSP = EGAUSSP + ABS(GAUSSP(I) - AGAUSSP(I))
  100          Write ( LOUT, 1002 )  I, XCELL, SQUARE(I), ASQUARE(I),
     &            CIRCLE(I), ACIRCLE(I), GAUSSP(I), AGAUSSP(I)

            Call CONSERVE ( SQUARE, 1, NX, CSQUARE )
            Call CONSERVE ( CIRCLE, 1, NX, CCIRCLE )
            Call CONSERVE ( GAUSSP, 1, NX, CGAUSSP )
            If ( ISTEP .eq. 1 ) Then
               ISQUARE = CSQUARE
               ICIRCLE = CCIRCLE
               IGAUSSP = CGAUSSP
            End If
            Write ( LOUT, 1003) CSQUARE, ISQUARE, CCIRCLE, ICIRCLE,
     &                          CGAUSSP, IGAUSSP
            ESQUARE = ESQUARE/CSQUARE
            ECIRCLE = ECIRCLE/CCIRCLE
            EGAUSSP = EGAUSSP/CGAUSSP
            Write ( LOUT, 1004) ESQUARE, ECIRCLE, EGAUSSP
         End If

c    Advance the densities one timestep using LCPFCT and CNVFCT . . .
C-----------------------------------------------------------------------
         If ( USE_LCP ) Then
            Call LCPFCT ( SQUARE, SQUARE, 1, NX, 0.,0.,0.,0., .true.)
            Call LCPFCT ( CIRCLE, CIRCLE, 1, NX, 0.,0.,0.,0., .true.)
            Call LCPFCT ( GAUSSP, GAUSSP, 1, NX, 0.,0.,0.,0., .true.)
         Else
            Call CNVFCT ( SQUARE, SQUARE, 1, NX, 0.,0.,0.,0., .true.)
            Call CNVFCT ( CIRCLE, CIRCLE, 1, NX, 0.,0.,0.,0., .true.)
            Call CNVFCT ( GAUSSP, GAUSSP, 1, NX, 0.,0.,0.,0., .true.)
         End If
         TIME = TIME + DT
```

```
 9999 Continue           !  End of the timestep loop.

      Stop
      End

C=======================================================================

      Subroutine PROFILE ( TYPE, TIME, ARRAY, XINT, NX, NXP, VX )

C-----------------------------------------------------------------------
c
c  This subroutine computes three different analytic density profiles
c  depending on the value of TYPE . . .
c
c     TYPE = 1   Square wave profile between 0.0 and HEIGHT
c     TYPE = 2   Semicircular (elliptical) profile from 0.0 to HEIGHT
c     TYPE = 3   Gaussian peak profile between 0.0 and HEIGHT
c
c  The profiles are presented on a periodic domain NX cells long and a
c  crude integration is done within each cell to better approximate the
c  curved functions and to give an analytic approximation accounting for
c  convection across a partial cell.
c
C-----------------------------------------------------------------------

      Implicit   NONE
      Integer    TYPE, NX, NXP, I, K
      Real       ARRAY(NX), XINT(NXP), TIME,  VX, SYSLEN, ARG
      Real       HEIGHT, X0, WIDTH,    XLEFT, XK, XRIGHT, XCENT
      Data       HEIGHT, X0, WIDTH / 1.0, 20.0, 10.0 /

      SYSLEN = XINT(NXP)
      Go To ( 100, 200, 300 ), TYPE

c  Compute the profile of the square wave . . .
C-----------------------------------------------------------------------
  100    XLEFT = (X0 - WIDTH) + VX*TIME
  101    If ( XLEFT .gt. SYSLEN ) Then
            XLEFT = XLEFT - SYSLEN
            Go To 101
         End If
  102    If ( XLEFT .lt. 0.0 ) Then
            XLEFT = XLEFT + SYSLEN
            Go To 102
         End If
         XRIGHT = XLEFT + 2.0*WIDTH

c  Loop over the cells in the numerical profile to be determined . . .
         Do 120 I = 1, NX
            ARRAY(I) = 0.0
            Do 110 K = 1, 10
               XK = XINT(I) + 0.1*(FLOAT(K)-0.5)*(XINT(I+1) - XINT(I))
               If ( XK .gt. XLEFT .and. XK .lt. XRIGHT ) Then
                  ARRAY(I) = ARRAY(I) + 0.1*HEIGHT
               Else
                  XK = XK + SYSLEN
                  If ( XK .gt. XLEFT .and. XK .lt. XRIGHT ) Then
                     ARRAY(I) = ARRAY(I) + 0.1*HEIGHT
                  End If
               End If
  110       Continue
```

```
  120    Continue
      Return

c  Compute the profile of the semicircle density hump . . .
C-------------------------------------------------------------------------
  200    XLEFT = (X0 - WIDTH) + VX*TIME
  201    If ( XLEFT .gt. SYSLEN ) Then
            XLEFT = XLEFT - SYSLEN
            Go To 201
         End If
         XRIGHT = XLEFT + 2.0*WIDTH
  202    If ( XLEFT .lt. 0.0 ) Then
            XLEFT = XLEFT + SYSLEN
            Go To 202
         End If
         XRIGHT = XLEFT + 2.0*WIDTH

c  Loop over the cells in the numerical profile to be determined . . .
         Do 220 I = 1, NX
            ARRAY(I) = 0.0
            Do 210 K = 1, 10
               XK = XINT(I) + 0.1*(FLOAT(K)-0.5)*(XINT(I+1) - XINT(I))
               If ( XK .gt. XLEFT .and. XK .lt. XRIGHT ) Then
                  XCENT = XLEFT + WIDTH
                  ARRAY(I) = ARRAY(I) + 0.1*HEIGHT*
     &               SQRT ( 1.0 - ((XK - XCENT)/WIDTH)**2 )
               Else
                  XK = XK + SYSLEN
                  If ( XK .gt. XLEFT .and. XK .lt. XRIGHT ) Then
                     XCENT = XLEFT + WIDTH
                     ARRAY(I) = ARRAY(I) + 0.1*HEIGHT*
     &                  SQRT ( 1.0 - ((XK - XCENT)/WIDTH)**2 )
                  End If
               End If
  210       Continue
  220    Continue
      Return

c  Compute the profile of the Gaussian density hump  . . .
C-------------------------------------------------------------------------
  300    XCENT = X0 + VX*TIME
  301    If ( XCENT .gt. SYSLEN ) Then
            XCENT = XCENT - SYSLEN
            Go To 301
         End If
  302    If ( XCENT .lt. 0.0 ) Then
            XCENT = XCENT + SYSLEN
            Go To 302
         End If

c  Loop over the cells in the numerical profile to be determined . . .
         Do 320 I = 1, NX
            ARRAY(I) = 0.0
            Do 310 K = 1, 10
               XK = XINT(I) + 0.1*(FLOAT(K)-0.5)*(XINT(I+1) - XINT(I))
               If ( XK .gt. (XCENT + 0.5*SYSLEN) ) XK = XK - SYSLEN
               If ( XK .lt. (XCENT - 0.5*SYSLEN) ) XK = XK + SYSLEN
               ARG = 4.0*((XK - XCENT)/WIDTH)**2
               ARRAY(I) = ARRAY(I) + 0.1*HEIGHT/EXP(AMIN1(30.0,ARG))
  310       Continue
  320    Continue
```

```
      Return
      End

C=====================================================================
```

LCPFCT Test #1 - Constant V Convection: step = 125 and TIME = 25.000
with DX = 1.000 DT = 0.200 and VX = 1.000

I	X(I)	Square	exact	Circle	exact	Gaussian	exact
1	0.50000	0.99997	1.00000	0.87922	0.83446	0.30138	0.29959
2	1.50000	0.99996	1.00000	0.83082	0.75899	0.18598	0.18597
3	2.50000	0.99973	1.00000	0.67214	0.66001	0.10652	0.10662
4	3.50000	0.94484	1.00000	0.43972	0.52391	0.05614	0.05645
5	4.50000	0.64565	1.00000	0.21917	0.29441	0.02694	0.02761
6	5.50000	0.30888	0.00000	0.07347	0.00000	0.01162	0.01247
7	6.50000	0.09164	0.00000	0.01098	0.00000	0.00444	0.00520
8	7.50000	0.00935	0.00000	0.00000	0.00000	0.00145	0.00200
9	8.50000	0.00000	0.00000	0.00000	0.00000	0.00034	0.00071
10	9.50000	0.00000	0.00000	0.00000	0.00000	0.00002	0.00023
11	10.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00007
12	11.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00002
13	12.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001
14	13.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
15	14.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
16	15.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
17	16.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
18	17.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
19	18.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
20	19.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
21	20.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
22	21.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
23	22.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
24	23.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
25	24.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
26	25.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
27	26.50000	0.00000	0.00000	0.00000	0.00000	0.00001	0.00000
28	27.50000	0.00000	0.00000	0.00000	0.00000	0.00002	0.00001
29	28.50000	0.00000	0.00000	0.00000	0.00000	0.00002	0.00002
30	29.50000	0.00000	0.00000	0.00000	0.00000	0.00003	0.00007
31	30.50000	0.00003	0.00000	0.00006	0.00000	0.00005	0.00023
32	31.50000	0.00004	0.00000	0.00013	0.00000	0.00199	0.00071
33	32.50000	0.00027	0.00000	0.00026	0.00000	0.00350	0.00200
34	33.50000	0.05516	0.00000	0.00122	0.00000	0.00380	0.00520
35	34.50000	0.35436	0.00000	0.07445	0.00000	0.00488	0.01247
36	35.50000	0.69113	1.00000	0.27283	0.29439	0.03055	0.02761
37	36.50000	0.90836	1.00000	0.48122	0.52390	0.08288	0.05645
38	37.50000	0.99065	1.00000	0.64240	0.66000	0.11505	0.10661
39	38.50000	1.00000	1.00000	0.75278	0.75899	0.15126	0.18596
40	39.50000	1.00000	1.00000	0.83335	0.83445	0.25564	0.29959
41	40.50000	1.00000	1.00000	0.89806	0.89245	0.44938	0.44576
42	41.50000	1.00000	1.00000	0.94672	0.93625	0.67349	0.61258
43	42.50000	1.00000	1.00000	0.97553	0.96779	0.83938	0.77751
44	43.50000	1.00000	1.00000	0.98617	0.98826	0.90622	0.91146
45	44.50000	1.00000	1.00000	0.98710	0.99833	0.91104	0.98686
46	45.50000	1.00000	1.00000	0.98710	0.99834	0.91104	0.98686
47	46.50000	1.00000	1.00000	0.98710	0.98826	0.91064	0.91147
48	47.50000	1.00000	1.00000	0.96931	0.96779	0.82071	0.77752
49	48.50000	1.00000	1.00000	0.90825	0.93625	0.64027	0.61259
50	49.50000	1.00000	1.00000	0.88015	0.89245	0.45563	0.44577
Conserved sums		20.00002	20.00000	15.70970	15.70969	8.86228	8.86227
Absolute error		0.08197		0.04005		0.05631	

LCPFCT Test #1 - Constant V Convection: step = 500 and TIME =100.000
with DX = 1.000 DT = 0.200 and VX = 1.000

I	X(I)	Square	exact	Circle	exact	Gaussian	exact
1	0.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	1.50000	0.00000	0.00000	0.00000	0.00000	0.00004	0.00000
3	2.50000	0.00000	0.00000	0.00000	0.00000	0.00011	0.00001
4	3.50000	0.00000	0.00000	0.00000	0.00000	0.00014	0.00002
5	4.50000	0.00005	0.00000	0.00000	0.00000	0.00014	0.00007
6	5.50000	0.00018	0.00000	0.00001	0.00000	0.00014	0.00023
7	6.50000	0.00065	0.00000	0.00001	0.00000	0.00019	0.00071
8	7.50000	0.00289	0.00000	0.00004	0.00000	0.00543	0.00200
9	8.50000	0.13042	0.00000	0.01387	0.00000	0.01946	0.00520
10	9.50000	0.39106	0.00000	0.11621	0.00000	0.02907	0.01247
11	10.50000	0.65869	1.00000	0.27572	0.29455	0.03172	0.02762
12	11.50000	0.85634	1.00000	0.44555	0.52397	0.03329	0.05647
13	12.50000	0.96248	1.00000	0.60186	0.66005	0.03789	0.10664
14	13.50000	0.99725	1.00000	0.73672	0.75902	0.12824	0.18600
15	14.50000	1.00000	1.00000	0.84563	0.83448	0.31265	0.29964
16	15.50000	1.00000	1.00000	0.92215	0.89247	0.53205	0.44582
17	16.50000	1.00000	1.00000	0.96265	0.93626	0.71844	0.61265
18	17.50000	1.00000	1.00000	0.97388	0.96780	0.83116	0.77758
19	18.50000	1.00000	1.00000	0.97423	0.98826	0.87123	0.91151
20	19.50000	1.00000	1.00000	0.97423	0.99834	0.87412	0.98687
21	20.50000	1.00000	1.00000	0.97275	0.99833	0.87412	0.98684
22	21.50000	1.00000	1.00000	0.96347	0.98825	0.87412	0.91142
23	22.50000	1.00000	1.00000	0.95660	0.96778	0.82358	0.77746
24	23.50000	1.00000	1.00000	0.95471	0.93623	0.66934	0.61252
25	24.50000	0.99995	1.00000	0.95370	0.89243	0.48613	0.44570
26	25.50000	0.99982	1.00000	0.92051	0.83443	0.32092	0.29954
27	26.50000	0.99935	1.00000	0.80082	0.75896	0.19435	0.18593
28	27.50000	0.99712	1.00000	0.61252	0.65996	0.10805	0.10659
29	28.50000	0.86959	1.00000	0.40154	0.52384	0.05424	0.05644
30	29.50000	0.60895	1.00000	0.21699	0.29425	0.02330	0.02760
31	30.50000	0.34132	0.00000	0.08919	0.00000	0.00746	0.01247
32	31.50000	0.14367	0.00000	0.02267	0.00000	0.00117	0.00520
33	32.50000	0.03753	0.00000	0.00152	0.00000	0.00000	0.00200
34	33.50000	0.00275	0.00000	0.00000	0.00000	0.00000	0.00071
35	34.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00023
36	35.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00007
37	36.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00002
38	37.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00001
39	38.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
40	39.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
41	40.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
42	41.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
43	42.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
44	43.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
45	44.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
46	45.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
47	46.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
48	47.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
49	48.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
50	49.50000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
Conserved sums		20.00005	20.00000	15.70975	15.70969	8.86231	8.86227
Absolute error		0.10505		0.06677		0.10650	

```
C=======================================================================

      Program SHOCK

C-----------------------------------------------------------------------
c
c     PROGRESSING 1D GASDYNAMIC SHOCK - LCPFCT Test # 2      August 1992
c
c     This program runs a simple 1D gasdynamic shock through a uniform
c     grid using LCPFCT and its utility routines. The fluid is ideal and
c     inviscid with constant GAMMA0 = 1.4.  The boundary conditions are
c     specified external values on both ends of the system.
c
C-----------------------------------------------------------------------

         Implicit  NONE

         Integer   NPT, ALPHA, BC1, BCN,   MAXSTP, IPRINT
         Parameter ( NPT = 202 )
         Integer   NX, NXP,    MX, ISTEP,  JSTEP, I,   LOUT
         Real      MACH,       V0,         DELTAX,     DELTAT
         Real      RHOSUM,     RHVSUM,     PRESUM,     ERGSUM
         Real      VNEW(NPT),  PNEW(NPT),  TNEW(NPT),  XINT(NPT)
         Real      CSAMB,      ERG_IN,     ERGAMB,     RELAX
         Real      TIME,       DTMAX

         Real      RHO_IN,     PRE_IN,     VEL_IN,     GAMMA0
         Real      RHOAMB,     PREAMB,     VELAMB,     GAMMAM
         Real      RHON(NPT),  RVXN(NPT),  RVTN(NPT),  ERGN(NPT)
         Common    / ARRAYS / RHON,   RVXN,   RVTN,   ERGN,   RELAX,
     &                        RHO_IN, PRE_IN, VEL_IN, GAMMA0,
     &                        RHOAMB, PREAMB, VELAMB, GAMMAM

 1000 Format ( '1', /, '    LCPFCT Test # 2 - Progressing Shock:',
     &         '   Step =', I4, '   NX =', I3, '   DT =', F6.3, / )
 1001 Format (2X, I3, 1P6E12.4)
 1002 Format ('    I    Density  Temperature   Pressure  ',
     &    '  Velocity     Energy    Interfaces', /)
 1003 Format ('0 Conservation Sums ', /, 5X, 1PE12.4, 12X, 3E12.4, /)

c  The Progressing Shock test run control parameters are specified here.
c  (change for other cases) . . .
C-----------------------------------------------------------------------
      MACH   = 5.0  ! Mach number of the incoming ambient flow
      V0     = 3.0  ! Shock speed in the lab frame
      DELTAX = 1.0  ! Cell size
      DELTAT = 0.05 ! Timestep for the calculation
      ALPHA  = 1    ! (1 = Cartesian, 2 = Cylindrical, 3 = Spherical)
      LOUT   =  11  ! Logical unit number of printed output device
      NX     =  50  ! Number of cells in the computational domain
      MX     =  10  ! Number of cells initialized behind the shock
      MAXSTP = 201  ! Maximum number of timesteps of length DELTAT
      IPRINT =  50  ! Frequency of intermediate result printouts

c  Initialize the variables in Common for use in GASDYN . . .
C-----------------------------------------------------------------------
      GAMMA0 = 1.4  ! Gas constant
      PREAMB = 1.0  ! Ambient (unshocked) pressure on the right
      RHOAMB = 1.0  ! Density of the unshocked fluid on the right
      RELAX  = 0.0  ! Relaxation rate, not used when BC1, BCN = 2
      GAMMAM = GAMMA0 - 1.0
```

```
c  The Rankine-Hugoniot conditions are set for boundaries . . .
C-----------------------------------------------------------------------
      CSAMB  = SQRT (GAMMA0*PREAMB/RHOAMB)
      VELAMB = -MACH*CSAMB
      VEL_IN = VELAMB*(GAMMAM + 2.0/MACH**2)/(GAMMA0 + 1.0)
      RHO_IN = RHOAMB*VELAMB/VEL_IN
      PRE_IN = PREAMB - RHO_IN*VEL_IN**2 + RHOAMB*VELAMB**2
      VELAMB = VELAMB + V0
      VEL_IN = VEL_IN + V0
      ERGAMB = PREAMB/(GAMMA0 - 1.0) + 0.5*RHOAMB*VELAMB**2
      ERG_IN = PRE_IN/(GAMMA0 - 1.0) + 0.5*RHO_IN*VEL_IN**2

c  Define the cell interface locations and physical variables . . .
C-----------------------------------------------------------------------
      NXP = NX + 1
      Do 10 I = 1, NXP
   10    XINT(I) = FLOAT(I-1)*DELTAX
      Do 20 I = MX+1, NX
         RHON(I) = RHOAMB
         RVXN(I) = RHOAMB*VELAMB
         RVTN(I) = 0.0
   20    ERGN(I) = ERGAMB
      Do 30 I = 1, MX
         RHON(I) = RHO_IN
         RVXN(I) = RHO_IN*VEL_IN
         RVTN(I) = 0.0
   30    ERGN(I) = ERG_IN

c  Begin loop over timesteps . . .
C-----------------------------------------------------------------------
      BC1 = 4
      BCN = 4
      Call RESIDIFF ( 1.000 )
      Call MAKEGRID ( XINT, XINT, 1, NXP, ALPHA )

      Do 9999 ISTEP = 1, MAXSTP

c  The results are printed when required . . .
C-----------------------------------------------------------------------
         If ( MOD(ISTEP-1, IPRINT) .eq. 0) Then

            JSTEP = ISTEP - 1
            Write ( LOUT, 1000 ) JSTEP, NX, DELTAT
            Do 40 I = 1, NX
               VNEW(I) = RVXN(I)/RHON(I)
               PNEW(I) = GAMMAM*(ERGN(I) - 0.5*RVXN(I)*VNEW(I))
   40          TNEW(I) = PNEW(I)/RHON(I)
            Write (  LOUT, 1002 )
            Write (  LOUT, 1001 ) ( I, RHON(I), TNEW(I), PNEW(I),
     &           VNEW(I), ERGN(I), XINT(I), I = 1, NX )
            Call CONSERVE (RHON, 1, NX, RHOSUM)
            Call CONSERVE (PNEW, 1, NX, PRESUM)
            Call CONSERVE (RVXN, 1, NX, RHVSUM)
            Call CONSERVE (ERGN, 1, NX, ERGSUM)
            PRESUM = PRESUM/GAMMAM
            Write (  LOUT, 1003 ) RHOSUM, PRESUM, RHVSUM, ERGSUM
         End If

c  The FCT integration of the continuity equations is performed . . .
C-----------------------------------------------------------------------
```

```
         Call GASDYN ( 1, NX, BC1, BCN, DELTAT )
         TIME = TIME + DELTAT

 9999 Continue    ! End of the loop over timesteps.

      Stop
      End

C=======================================================================

      Subroutine GASDYN ( K1, KN, BC1, BCN, DT )

C-----------------------------------------------------------------------
c
c  This routine integrates the gasdynamic equations using the momentum
c  component RVRN as the direction of integration and the momentum RVTN
c  as the transverse direction.  In 2D models the two directions of
c  integration are chosen by exchanging RVRN and RVTN in Common.

c  K1  . . .   Index of the integration's first cell
c  KN  . . .   Index of the integration's last cell
c  BC1 . . .   Indicates boundary condition on integration at K1
c  BCN . . .   Indicates boundary condition on integration at K1
c  DT  . . .   Timestep for the integrations of this step
c
C-----------------------------------------------------------------------

         Implicit  NONE
         Integer   NPT, K1, K1P, BC1, BCN, K, KN, KNP, IT
         Parameter ( NPT = 202 )
         Logical   PBC
         Real      SBC1,  SRV1,  SBCN,     SRVN,  VRHO1, VRHON
         Real      VRVR1, VRVRN, VRVT1,    VRVTN, VERG1, VERGN
         Real      MPINT(NPT), VEL(NPT),   UNIT(NPT),  ZERO(NPT)
         Real      RHOO(NPT),  RVRO(NPT),  RVTO(NPT),  ERGO(NPT)
         Real      VINT(NPT),  PRE(NPT),   MPVINT(NPT)
         Real      DTSUB,      DT,         RELAX
         Data      UNIT / NPT*1.0 /,       ZERO / NPT*0.0 /

         Real      RHO_IN,     PRE_IN,     VEL_IN,     GAMMA0
         Real      RHOAMB,     PREAMB,     VELAMB,     GAMMAM
         Real      RHON(NPT),  RVRN(NPT),  RVTN(NPT),  ERGN(NPT)
         Common    / ARRAYS / RHON,   RVRN,   RVTN,   ERGN,   RELAX,
     &                        RHO_IN, PRE_IN, VEL_IN, GAMMA0,
     &                        RHOAMB, PREAMB, VELAMB, GAMMAM

c  Prepare for the time integration. Index K is either I or J depending
c  on the definitions of RVRN and RVTN. Copies of the physical variable
c  are needed to recover values for the whole step integration . . .
C-----------------------------------------------------------------------
      KNP = KN + 1
      K1P = K1 + 1
      PBC = .false.
      If ( BC1.eq.3 .OR. BCN.eq.3 ) PBC = .true.
      Do 50 K = K1, KN
         RHOO(K) = RHON(K)
         RVRO(K) = RVRN(K)
         RVTO(K) = RVTN(K)
   50    ERGO(K) = ERGN(K)

c  Integrate first the half step then the whole step . . .
```

```
C-------------------------------------------------------------------------
      Do 500 IT = 1, 2
         DTSUB = 0.5*DT*FLOAT(IT)

         Do 100 K = K1, KN
            VEL(K) = RVRN(K)/RHON(K)
  100       PRE(K) = GAMMAM*(ERGN(K)
     &              - 0.5*(RVRN(K)**2 + RVTN(K)**2)/RHON(K))

c  Calculate the interface velocities and pressures as weighted values
c  of the cell-centered values computed just above . . .
C-------------------------------------------------------------------------
         Do 200 K = K1+1, KN
            MPVINT(K) = 1.0/( RHON(K) + RHON(K-1) )
            VINT(K)   =  (VEL(K)*RHON(K-1) + VEL(K-1)*RHON(K))*MPVINT(K)
            MPINT(K)  = -(PRE(K)*RHON(K-1) + PRE(K-1)*RHON(K))*MPVINT(K)
  200       MPVINT(K) = -( PRE(K)*VEL(K)*RHON(K-1)
     &                  + PRE(K-1)*VEL(K-1)*RHON(K) )*MPVINT(K)

c  The unweighted interface averages can be computed as follows . . .
c           VINT(K)  =  0.5*(VEL(K) + VEL(K-1))
c           MPINT(K) = -0.5*(PRE(K) + PRE(K-1))
c  200      MPVINT(K) = MPINT(K)*VINT(K)

c  Call the FCT utility routines and set the boundary conditions. Other
c  boundary conditions could be added for inflow, outflow, etc . . .
c  BC1, BCN = 1  => ideal solid wall or axis boundary condition
c  BC1, BCN = 2  => an extrapolative outflow boundary condition
c  BC1, BCN = 3  => periodic boundary conditions . . .
c  BC1, BCN = 4  => specified boundary values (e.g. shock tube problem)
C-------------------------------------------------------------------------
         Go To ( 310, 320, 330, 340 ), BC1
  310       VINT(K1)   = 0.0
            MPINT(K1)  = - PRE(K1)
            MPVINT(K1) = 0.0
            Go To 350
  320       VINT(K1)   = VEL(K1)*(1.0 - RELAX)
            MPINT(K1)  = - PRE(K1)*(1.0 - RELAX) - RELAX*PRE_IN
            MPVINT(K1) = MPINT(K1)*VINT(K1)
            Go To 350
  330       MPVINT(K1) = 1.0/( RHON(K1) + RHON(KN) )
            VINT(K1)   = (VEL(K1)*RHON(KN)+VEL(KN)*RHON(K1)) *MPVINT(K1)
            MPINT(K1)  = -(PRE(K1)*RHON(KN)+PRE(KN)*RHON(K1))*MPVINT(K1)
            MPVINT(K1) = -(PRE(K1)*VEL(K1)*RHON(KN)
     &                  + PRE(KN)*VEL(KN)*RHON(K1))*MPVINT(K1)
            Go To 350
  340       VINT(K1)   = VEL_IN
            MPINT(K1)  = - PRE_IN
            MPVINT(K1) = - PRE_IN*VEL_IN

  350    Go To ( 410, 420, 430, 440 ), BCN
  410       VINT(KNP)   = 0.0
            MPINT(KNP)  = - PRE(KN)
            MPVINT(KNP) = 0.0
            Go To 450
  420       VINT(KNP)   = VEL(KN)*(1.0 - RELAX)
            MPINT(KNP)  = - PRE(KN)*(1.0 - RELAX) - RELAX*PREAMB
            MPVINT(KNP) = MPINT(KNP)*VINT(KNP)
            Go To 450
  430       VINT(KNP)   = VINT(K1)
            MPINT(KNP)  = MPINT(K1)
```

```
            MPVINT(KNP) = MPVINT(K1)
            Go To 450
  440       VINT(KNP)   = VELAMB
            MPINT(KNP)  = - PREAMB
            MPVINT(KNP) = - PREAMB*VELAMB
  450    Continue

c  The velocity dependent FCT coefficients are set and the boundary
c  condition calculations are completed.  Here the periodic boundary
c  conditions require no action as (S)lope and (V)alue boundary value
c  specifiers are ignored in LCPFCT when PBC = .true.
C-------------------------------------------------------------------------
         Call VELOCITY ( VINT, K1, KNP, DTSUB )

         Go To ( 510, 520, 550, 540 ), BC1
  510       Call ZEROFLUX ( K1 )
            SBC1  = 1.0
            SRV1  = -1.0
            VRHO1 = 0.0
            VRVR1 = 0.0
            VRVT1 = 0.0
            VERG1 = 0.0
            Go To 550
  520       Call ZERODIFF ( K1 )
            SBC1  = 1.0 - RELAX
            SRV1  = 1.0 - RELAX
            VRHO1 = RELAX*RHO_IN
            VRVR1 = 0.0
            VRVT1 = 0.0
            VERG1 = RELAX*PRE_IN/GAMMAM
            Go To 550
  540       SBC1  = 0.0
            SRV1  = 0.0
            VRHO1 = RHO_IN
            VRVR1 = RHO_IN*VEL_IN
            VRVT1 = 0.0
            VERG1 = PRE_IN/GAMMAM + 0.5*RHO_IN*VEL_IN**2

  550    Go To ( 610, 620, 650, 640 ), BCN
  610       Call ZEROFLUX ( KNP )
            SBCN = 1.0
            SRVN = -1.0
            VRHON = 0.0
            VRVRN = 0.0
            VRVTN = 0.0
            VERGN = 0.0
            Go To 650
  620       Call ZERODIFF ( KNP )
            SBCN  = 1.0 - RELAX
            SRVN  = 1.0 - RELAX
            VRHON = RELAX*RHOAMB
            VRVRN = 0.0
            VRVTN = 0.0
            VERGN = RELAX*PREAMB/GAMMAM
            Go To 650
  640       SBCN  = 0.0
            SRVN  = 0.0
            VRHON = RHOAMB
            VRVRN = RHOAMB*VELAMB
            VRVTN = 0.0
            VERGN = PREAMB/GAMMAM + 0.5*RHOAMB*VELAMB**2
```

```
  650    Continue

c  Integrate the continuity equations using LCPFCT . . .
C-------------------------------------------------------------------------
         Call LCPFCT ( RHOO, RHON, K1,KN, SBC1,VRHO1, SBCN,VRHON, PBC )

         Call SOURCES( K1,KN, DTSUB, 5, UNIT, MPINT,
     &                                         MPINT(K1),  MPINT(KNP)  )

         Call LCPFCT ( RVRO, RVRN, K1,KN, SRV1,VRVR1, SRVN,VRVRN, PBC )

         Call LCPFCT ( RVTO, RVTN, K1,KN, SBC1,VRVT1, SBCN,VRVTN, PBC )

         Call SOURCES( K1,KN, DTSUB, 4, UNIT, MPVINT,
     &                                         MPVINT(K1), MPVINT(KNP) )

         Call LCPFCT ( ERGO, ERGN, K1,KN, SBC1,VERG1, SBCN,VERGN, PBC )

  500 Continue       ! End of halfstep-wholestep loop.
      Return
      End

C=========================================================================
```

LCPFCT Test # 2 - Progressing Shock: Step = 0 NX = 50 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces
1	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	0.0000E+00
2	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	1.0000E+00
3	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	2.0000E+00
4	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	3.0000E+00
5	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	4.0000E+00
6	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	5.0000E+00
7	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	6.0000E+00
8	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	7.0000E+00
9	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	8.0000E+00
10	5.0000E+00	5.8000E+00	2.9000E+01	1.8168E+00	8.0752E+01	9.0000E+00
11	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.0000E+01
12	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.1000E+01
13	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.2000E+01
14	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.3000E+01
15	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.4000E+01
16	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.5000E+01
17	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.6000E+01
18	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.7000E+01
19	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.8000E+01
20	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	1.9000E+01
21	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.0000E+01
22	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.1000E+01
23	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.2000E+01
24	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.3000E+01
25	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.4000E+01
26	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.5000E+01
27	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.6000E+01
28	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.7000E+01
29	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.8000E+01
30	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	2.9000E+01
31	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.0000E+01
32	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.1000E+01
33	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.2000E+01
34	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.3000E+01
35	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.4000E+01
36	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.5000E+01
37	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.6000E+01
38	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.7000E+01
39	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.8000E+01
40	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.9000E+01
41	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.0000E+01
42	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.1000E+01
43	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.2000E+01
44	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.3000E+01
45	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.4000E+01
46	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.5000E+01
47	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.6000E+01
48	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.7000E+01
49	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.8000E+01
50	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.9000E+01

Conservation Sums

9.0000E+01		8.2500E+02	-2.5804E+01	1.0776E+03

LCPFCT Test # 2 - Progressing Shock: Step = 150 NX = 50 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces
1	4.9983E+00	5.7992E+00	2.8986E+01	1.8177E+00	8.0724E+01	0.0000E+00
2	4.9983E+00	5.7992E+00	2.8986E+01	1.8177E+00	8.0723E+01	1.0000E+00
3	4.9980E+00	5.7991E+00	2.8984E+01	1.8179E+00	8.0719E+01	2.0000E+00
4	4.9974E+00	5.7988E+00	2.8979E+01	1.8183E+00	8.0708E+01	3.0000E+00
5	4.9969E+00	5.7986E+00	2.8975E+01	1.8186E+00	8.0701E+01	4.0000E+00
6	4.9962E+00	5.7983E+00	2.8970E+01	1.8189E+00	8.0689E+01	5.0000E+00
7	4.9958E+00	5.7980E+00	2.8966E+01	1.8191E+00	8.0680E+01	6.0000E+00
8	4.9953E+00	5.7979E+00	2.8962E+01	1.8194E+00	8.0674E+01	7.0000E+00
9	4.9949E+00	5.7977E+00	2.8958E+01	1.8197E+00	8.0666E+01	8.0000E+00
10	4.9946E+00	5.7973E+00	2.8955E+01	1.8199E+00	8.0659E+01	9.0000E+00
11	4.9941E+00	5.7973E+00	2.8952E+01	1.8201E+00	8.0653E+01	1.0000E+01
12	4.9937E+00	5.7972E+00	2.8950E+01	1.8203E+00	8.0648E+01	1.1000E+01
13	4.9936E+00	5.7969E+00	2.8947E+01	1.8205E+00	8.0643E+01	1.2000E+01
14	4.9933E+00	5.7967E+00	2.8945E+01	1.8206E+00	8.0638E+01	1.3000E+01
15	4.9929E+00	5.7969E+00	2.8944E+01	1.8208E+00	8.0636E+01	1.4000E+01
16	4.9929E+00	5.7967E+00	2.8942E+01	1.8208E+00	8.0632E+01	1.5000E+01
17	4.9929E+00	5.7966E+00	2.8942E+01	1.8209E+00	8.0632E+01	1.6000E+01
18	4.9929E+00	5.7966E+00	2.8942E+01	1.8209E+00	8.0632E+01	1.7000E+01
19	4.9929E+00	5.7966E+00	2.8942E+01	1.8208E+00	8.0632E+01	1.8000E+01
20	4.9931E+00	5.7968E+00	2.8944E+01	1.8208E+00	8.0637E+01	1.9000E+01
21	4.9932E+00	5.7969E+00	2.8945E+01	1.8206E+00	8.0638E+01	2.0000E+01
22	4.9932E+00	5.7976E+00	2.8949E+01	1.8203E+00	8.0645E+01	2.1000E+01
23	4.9937E+00	5.7983E+00	2.8955E+01	1.8201E+00	8.0659E+01	2.2000E+01
24	4.9938E+00	5.7989E+00	2.8958E+01	1.8196E+00	8.0663E+01	2.3000E+01
25	4.9946E+00	5.7994E+00	2.8966E+01	1.8191E+00	8.0678E+01	2.4000E+01
26	4.9946E+00	5.8013E+00	2.8975E+01	1.8186E+00	8.0697E+01	2.5000E+01
27	4.9946E+00	5.8018E+00	2.8977E+01	1.8186E+00	8.0703E+01	2.6000E+01
28	4.9948E+00	5.8020E+00	2.8980E+01	1.8178E+00	8.0702E+01	2.7000E+01
29	4.9954E+00	5.8049E+00	2.8998E+01	1.8173E+00	8.0744E+01	2.8000E+01
30	4.9954E+00	5.8072E+00	2.9009E+01	1.8172E+00	8.0771E+01	2.9000E+01
31	4.9954E+00	5.8074E+00	2.9010E+01	1.8171E+00	8.0773E+01	3.0000E+01
32	4.9948E+00	5.8091E+00	2.9015E+01	1.8159E+00	8.0773E+01	3.1000E+01
33	3.1692E+00	5.6326E+00	1.7851E+01	9.4539E-01	4.6043E+01	3.2000E+01
34	9.9995E-01	9.9907E-01	9.9902E-01	-2.9166E+00	6.7506E+00	3.3000E+01
35	9.9981E-01	9.9897E-01	9.9878E-01	-2.9170E+00	6.7506E+00	3.4000E+01
36	9.9981E-01	9.9950E-01	9.9931E-01	-2.9166E+00	6.7507E+00	3.5000E+01
37	9.9993E-01	1.0001E+00	1.0000E+00	-2.9162E+00	6.7518E+00	3.6000E+01
38	1.0000E+00	1.0000E+00	1.0000E+00	-2.9160E+00	6.7518E+00	3.7000E+01
39	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.8000E+01
40	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	3.9000E+01
41	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.0000E+01
42	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.1000E+01
43	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.2000E+01
44	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.3000E+01
45	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.4000E+01
46	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.5000E+01
47	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.6000E+01
48	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.7000E+01
49	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.8000E+01
50	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.9000E+01

Conservation Sums

1.8000E+02	2.4044E+03	2.4420E+02	2.7426E+03	

LCPFCT Test # 2 - Progressing Shock: Step = 200 NX = 50 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces

1	4.9968E+00	5.7985E+00	2.8974E+01	1.8186E+00	8.0698E+01	0.0000E+00
2	4.9967E+00	5.7985E+00	2.8973E+01	1.8186E+00	8.0696E+01	1.0000E+00
3	4.9966E+00	5.7985E+00	2.8973E+01	1.8187E+00	8.0696E+01	2.0000E+00
4	4.9959E+00	5.7981E+00	2.8967E+01	1.8191E+00	8.0683E+01	3.0000E+00
5	4.9956E+00	5.7980E+00	2.8964E+01	1.8193E+00	8.0678E+01	4.0000E+00
6	4.9951E+00	5.7977E+00	2.8960E+01	1.8196E+00	8.0669E+01	5.0000E+00
7	4.9948E+00	5.7976E+00	2.8958E+01	1.8198E+00	8.0665E+01	6.0000E+00
8	4.9942E+00	5.7974E+00	2.8953E+01	1.8200E+00	8.0655E+01	7.0000E+00
9	4.9939E+00	5.7972E+00	2.8951E+01	1.8202E+00	8.0650E+01	8.0000E+00
10	4.9937E+00	5.7970E+00	2.8948E+01	1.8204E+00	8.0645E+01	9.0000E+00
11	4.9932E+00	5.7970E+00	2.8946E+01	1.8206E+00	8.0640E+01	1.0000E+01
12	4.9931E+00	5.7968E+00	2.8944E+01	1.8207E+00	8.0636E+01	1.1000E+01
13	4.9930E+00	5.7967E+00	2.8943E+01	1.8208E+00	8.0634E+01	1.2000E+01
14	4.9929E+00	5.7967E+00	2.8943E+01	1.8208E+00	8.0634E+01	1.3000E+01
15	4.9929E+00	5.7967E+00	2.8943E+01	1.8208E+00	8.0634E+01	1.4000E+01
16	4.9930E+00	5.7967E+00	2.8943E+01	1.8208E+00	8.0634E+01	1.5000E+01
17	4.9930E+00	5.7968E+00	2.8943E+01	1.8208E+00	8.0634E+01	1.6000E+01
18	4.9932E+00	5.7969E+00	2.8945E+01	1.8207E+00	8.0638E+01	1.7000E+01
19	4.9936E+00	5.7969E+00	2.8947E+01	1.8204E+00	8.0642E+01	1.8000E+01
20	4.9940E+00	5.7973E+00	2.8952E+01	1.8201E+00	8.0652E+01	1.9000E+01
21	4.9947E+00	5.7974E+00	2.8957E+01	1.8198E+00	8.0662E+01	2.0000E+01
22	4.9955E+00	5.7980E+00	2.8964E+01	1.8194E+00	8.0677E+01	2.1000E+01
23	4.9963E+00	5.7981E+00	2.8969E+01	1.8188E+00	8.0686E+01	2.2000E+01
24	4.9971E+00	5.7987E+00	2.8977E+01	1.8185E+00	8.0704E+01	2.3000E+01
25	4.9981E+00	5.7990E+00	2.8984E+01	1.8181E+00	8.0720E+01	2.4000E+01
26	4.9981E+00	5.7996E+00	2.8987E+01	1.8175E+00	8.0723E+01	2.5000E+01
27	4.9988E+00	5.8005E+00	2.8995E+01	1.8171E+00	8.0741E+01	2.6000E+01
28	4.9996E+00	5.8007E+00	2.9002E+01	1.8168E+00	8.0755E+01	2.7000E+01
29	4.9997E+00	5.8020E+00	2.9008E+01	1.8163E+00	8.0768E+01	2.8000E+01
30	4.9996E+00	5.8025E+00	2.9010E+01	1.8158E+00	8.0767E+01	2.9000E+01
31	4.9996E+00	5.8039E+00	2.9017E+01	1.8156E+00	8.0784E+01	3.0000E+01
32	4.9996E+00	5.8050E+00	2.9022E+01	1.8152E+00	8.0793E+01	3.1000E+01
33	4.9990E+00	5.8061E+00	2.9025E+01	1.8153E+00	8.0798E+01	3.2000E+01
34	4.9987E+00	5.8069E+00	2.9027E+01	1.8148E+00	8.0800E+01	3.3000E+01
35	4.9980E+00	5.8080E+00	2.9028E+01	1.8147E+00	8.0800E+01	3.4000E+01
36	4.9979E+00	5.8084E+00	2.9030E+01	1.8147E+00	8.0804E+01	3.5000E+01
37	4.9977E+00	5.8088E+00	2.9030E+01	1.8148E+00	8.0805E+01	3.6000E+01
38	4.9977E+00	5.8083E+00	2.9028E+01	1.8153E+00	8.0805E+01	3.7000E+01
39	4.9942E+00	5.8118E+00	2.9025E+01	1.7854E+00	8.0522E+01	3.8000E+01
40	4.9107E+00	5.8264E+00	2.8612E+01	1.8047E+00	7.9527E+01	3.9000E+01
41	1.2490E+00	2.4009E+00	2.9988E+00	-2.0546E+00	1.0133E+01	4.0000E+01
42	9.9955E-01	9.9660E-01	9.9616E-01	-2.9194E+00	6.7498E+00	4.1000E+01
43	9.9955E-01	9.9832E-01	9.9788E-01	-2.9179E+00	6.7498E+00	4.2000E+01
44	9.9971E-01	9.9942E-01	9.9912E-01	-2.9168E+00	6.7505E+00	4.3000E+01
45	1.0000E+00	1.0001E+00	1.0001E+00	-2.9160E+00	6.7518E+00	4.4000E+01
46	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.5000E+01
47	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.6000E+01
48	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.7000E+01
49	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.8000E+01
50	1.0000E+00	1.0000E+00	1.0000E+00	-2.9161E+00	6.7518E+00	4.9000E+01

Conservation Sums

2.1000E+02		2.9269E+03	3.3419E+02	3.2976E+03

```
C=====================================================================

      Program DIAPHRAGM

C---------------------------------------------------------------------
c
c     1D BURSTING DIAPHRAGM PROBLEM - LCPFCT Test # 3        August 1992
c
c  This program runs a very simple 1D bursting diaphragm test problem
c  using LCPFCT and its utility routines.  The fluid is ideal and
c  inviscid with constant GAMMA = 1.667.  The end walls are reflecting
c  so the fluid should be totally contained in the domain.
c
C---------------------------------------------------------------------

         Implicit  NONE

         Integer   NPT, ALPHA, BC1, BCN,   MAXSTP,     IPRINT
         Parameter ( NPT = 202 )
         Integer   NX, NXP,    MX, ISTEP,  JSTEP, I,   LOUT
         Real      TIME,       DELTAX,     DELTAT,     DX_JIGGLE
         REAL      XGRID(NPT), XNEXT(NPT), DXOFF,      DX_OSC
         Real      RHOSUM,     RHVSUM,     PRESUM,     ERGSUM
         Real      VNEW(NPT),  PNEW(NPT),  TNEW(NPT)
         Real      VELX,       VXPAND,     SCALEG
         Real      ERG_IN,     ERGAMB,     RELAX

         Real      RHO_IN,     PRE_IN,     VEL_IN,     GAMMA0
         Real      RHOAMB,     PREAMB,     VELAMB,     GAMMAM
         Real      RHON(NPT),  RVXN(NPT),  RVTN(NPT),  ERGN(NPT)
         Common    / ARRAYS / RHON,   RVXN,   RVTN,   ERGN,   RELAX,
     &                        RHO_IN, PRE_IN, VEL_IN, GAMMA0,
     &                        RHOAMB, PREAMB, VELAMB, GAMMAM

 1000 Format ( '1', /, '    LCPFCT Test # 3 - Bursting Diaphragm:',
     &         '   Step =', I4, '   NX =', I3, '   DT =', F6.3, / )
 1001 Format ( 2X, I3, 6F12.5 )
 1002 Format ( '    I    Density  Temperature   Pressure  ',
     &    '  Velocity     Energy    Interfaces', / )
 1003 Format ( '0 Conservation Sums ', /, 5X, F12.5, 12X, 3F12.5, / )

c  The Bursting Diaphragm run control parameters are specified here.
c  (change for other cases) . . .
C---------------------------------------------------------------------
      DELTAX = 1.0  ! Cell size
      DELTAT = 0.05 ! Timestep for the calculation
      ALPHA  =   1  ! (1 = Cartesian, 2 = Cylindrical, 3 = Spherical)
      LOUT   =  11  ! Logical unit number of printed output device
      NX     = 100  ! Number of cells in the computational domain
      MX     =  60  ! Number of cells initialized behind the shock
      MAXSTP = 1601 ! Maximum number of timesteps of length DELTAT
      IPRINT =  50  ! Frequency of intermediate result printouts

c  Initialize the variables in Common for use in GASDYN . . .
C---------------------------------------------------------------------
      RHO_IN =  1.0    ! Initial mass density behind the diaphragm
      PRE_IN = 10.0    ! Initial (higher) pressure behind the diaphragm
      VEL_IN =  0.0    ! Initial velocity
      VXPAND =  3.1    ! Characteristic system expansion velocity
      DX_OSC =  0.125  ! Amplitude of the grid jiggling after step 200
      GAMMA0 = 1.66667 ! Gas constant
```

```
      PREAMB =  1.0    ! Ambient (unshocked) pressure on the right
      RHOAMB =  1.0    ! Density of the unshocked fluid on the right
      VELAMB =  0.0    ! Initial velocity
      RELAX  =  0.002  ! Relaxation rate, used when BC1, BCN = 2
      GAMMAM = GAMMA0 - 1.0
      ERGAMB = PREAMB/(GAMMA0 - 1.0) + 0.5*RHOAMB*VELAMB**2
      ERG_IN = PRE_IN/(GAMMA0 - 1.0) + 0.5*RHO_IN*VEL_IN**2

c  Set up the fluid variables with the diaphragm at interface MX+1 . . .
C-------------------------------------------------------------------------
      Do 10 I = MX+1, NX
         RHON(I) = RHOAMB
         RVXN(I) = RHOAMB*VELAMB
         RVTN(I) = 0.0
   10    ERGN(I) = ERGAMB
      Do 20 I = 1, MX
         RHON(I) = RHO_IN
         RVXN(I) = RHO_IN*VEL_IN
         RVTN(I) = 0.0
   20    ERGN(I) = ERG_IN

c  Begin loop over timesteps . . .
C-------------------------------------------------------------------------
      BC1 = 1
      BCN = 1
      Call RESIDIFF ( 0.998 )

      TIME = 0.0
      Do 9999 ISTEP = 1, MAXSTP

c  Define the cell interface locations and physical variables.  The grid
c  is expanded at the rate VELX at I = NXP after step 201 (as the shock
c  approaches the boundary) by making XNEXT at the end of the timestep
c  proportionately larger than XGRID at the beginning of the step.  The
c  system length is then renormalized to its original size, capturing
c  the similarity solution by equating the grid expansion to the shock
c  velocity.  A small jiggle is added to this systematic expansion to
c  show the added flexibility of the continuity solver.
C-------------------------------------------------------------------------
         NXP = NX + 1
         If ( ISTEP .gt. 200 ) Then
            VELX = VXPAND
            DXOFF = DX_OSC
            If ( MOD(ISTEP,2) .eq. 0 ) DXOFF = - DX_OSC
         Else
            VELX = 0.0
            DXOFF = 0.0
         End If

         Do 30 I = 1, NXP
            XGRID(I)  = FLOAT(I-MX-1)*DELTAX
   30       XNEXT(I) = XGRID(I)
         SCALEG = (VELX*DELTAT + XGRID(NXP))/XGRID(NXP)
         Do 40 I = 3, NX-2
            XNEXT(I)   = (XNEXT(I) + DXOFF)*SCALEG
  40        XGRID(I)   = XGRID(I) - DXOFF

         Call MAKEGRID ( XGRID, XNEXT, 1, NXP, ALPHA )

c  The results are printed when required . . .
C-------------------------------------------------------------------------
```

```
         If ( MOD(ISTEP-1, IPRINT) .eq. 0) Then
            JSTEP = ISTEP - 1
            IPRINT = 2*(ISTEP - 1)
            If ( ISTEP .lt. 200 ) IPRINT = 50
            Write ( LOUT, 1000 ) JSTEP, NX, DELTAT
            Do 50 I = 1, NX
               VNEW(I) = RVXN(I)/RHON(I)
               PNEW(I) = GAMMAM*(ERGN(I) - 0.5*RVXN(I)*VNEW(I))
   50          TNEW(I) = PNEW(I)/RHON(I)
            Write (  LOUT, 1002 )
            Write (  LOUT, 1001 ) ( I, RHON(I), TNEW(I), PNEW(I),
     &           VNEW(I), ERGN(I), XGRID(I), I = 1, NX )
            Call CONSERVE (RHON, 1, NX, RHOSUM)
            Call CONSERVE (PNEW, 1, NX, PRESUM)
            Call CONSERVE (RVXN, 1, NX, RHVSUM)
            Call CONSERVE (ERGN, 1, NX, ERGSUM)
            Write (  LOUT, 1003 ) RHOSUM, PRESUM, RHVSUM, ERGSUM
         End If

c  The FCT integration of the continuity equations is performed . . .
C-------------------------------------------------------------------------
         Call GASDYN ( 1, NX, BC1, BCN, DELTAT )
         TIME = TIME + DELTAT

 9999 Continue    ! End of the loop over timesteps.

      Stop
      End

C=========================================================================
```

LCPFCT Test # 3 - Bursting Diaphragm: Step = 0 NX =100 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces
1	1.00000	10.00000	10.00000	0.00000	14.99993	-60.00000
2	1.00000	10.00000	10.00000	0.00000	14.99993	-59.00000
3	1.00000	10.00000	10.00000	0.00000	14.99993	-58.00000
4	1.00000	10.00000	10.00000	0.00000	14.99993	-57.00000
5	1.00000	10.00000	10.00000	0.00000	14.99993	-56.00000
6	1.00000	10.00000	10.00000	0.00000	14.99993	-55.00000
7	1.00000	10.00000	10.00000	0.00000	14.99993	-54.00000
8	1.00000	10.00000	10.00000	0.00000	14.99993	-53.00000
9	1.00000	10.00000	10.00000	0.00000	14.99993	-52.00000
10	1.00000	10.00000	10.00000	0.00000	14.99993	-51.00000
11	1.00000	10.00000	10.00000	0.00000	14.99993	-50.00000
12	1.00000	10.00000	10.00000	0.00000	14.99993	-49.00000
13	1.00000	10.00000	10.00000	0.00000	14.99993	-48.00000
14	1.00000	10.00000	10.00000	0.00000	14.99993	-47.00000
15	1.00000	10.00000	10.00000	0.00000	14.99993	-46.00000
16	1.00000	10.00000	10.00000	0.00000	14.99993	-45.00000
17	1.00000	10.00000	10.00000	0.00000	14.99993	-44.00000
18	1.00000	10.00000	10.00000	0.00000	14.99993	-43.00000
19	1.00000	10.00000	10.00000	0.00000	14.99993	-42.00000
20	1.00000	10.00000	10.00000	0.00000	14.99993	-41.00000
21	1.00000	10.00000	10.00000	0.00000	14.99993	-40.00000
22	1.00000	10.00000	10.00000	0.00000	14.99993	-39.00000
23	1.00000	10.00000	10.00000	0.00000	14.99993	-38.00000
24	1.00000	10.00000	10.00000	0.00000	14.99993	-37.00000
25	1.00000	10.00000	10.00000	0.00000	14.99993	-36.00000
26	1.00000	10.00000	10.00000	0.00000	14.99993	-35.00000
27	1.00000	10.00000	10.00000	0.00000	14.99993	-34.00000
28	1.00000	10.00000	10.00000	0.00000	14.99993	-33.00000
29	1.00000	10.00000	10.00000	0.00000	14.99993	-32.00000
30	1.00000	10.00000	10.00000	0.00000	14.99993	-31.00000
31	1.00000	10.00000	10.00000	0.00000	14.99993	-30.00000
32	1.00000	10.00000	10.00000	0.00000	14.99993	-29.00000
33	1.00000	10.00000	10.00000	0.00000	14.99993	-28.00000
34	1.00000	10.00000	10.00000	0.00000	14.99993	-27.00000
35	1.00000	10.00000	10.00000	0.00000	14.99993	-26.00000
36	1.00000	10.00000	10.00000	0.00000	14.99993	-25.00000
37	1.00000	10.00000	10.00000	0.00000	14.99993	-24.00000
38	1.00000	10.00000	10.00000	0.00000	14.99993	-23.00000
39	1.00000	10.00000	10.00000	0.00000	14.99993	-22.00000
40	1.00000	10.00000	10.00000	0.00000	14.99993	-21.00000
41	1.00000	10.00000	10.00000	0.00000	14.99993	-20.00000
42	1.00000	10.00000	10.00000	0.00000	14.99993	-19.00000
43	1.00000	10.00000	10.00000	0.00000	14.99993	-18.00000
44	1.00000	10.00000	10.00000	0.00000	14.99993	-17.00000
45	1.00000	10.00000	10.00000	0.00000	14.99993	-16.00000
46	1.00000	10.00000	10.00000	0.00000	14.99993	-15.00000
47	1.00000	10.00000	10.00000	0.00000	14.99993	-14.00000
48	1.00000	10.00000	10.00000	0.00000	14.99993	-13.00000
49	1.00000	10.00000	10.00000	0.00000	14.99993	-12.00000
50	1.00000	10.00000	10.00000	0.00000	14.99993	-11.00000
51	1.00000	10.00000	10.00000	0.00000	14.99993	-10.00000
52	1.00000	10.00000	10.00000	0.00000	14.99993	-9.00000
53	1.00000	10.00000	10.00000	0.00000	14.99993	-8.00000
54	1.00000	10.00000	10.00000	0.00000	14.99993	-7.00000
55	1.00000	10.00000	10.00000	0.00000	14.99993	-6.00000
56	1.00000	10.00000	10.00000	0.00000	14.99993	-5.00000
57	1.00000	10.00000	10.00000	0.00000	14.99993	-4.00000

58	1.00000	10.00000	10.00000	0.00000	14.99993	-3.00000
59	1.00000	10.00000	10.00000	0.00000	14.99993	-2.00000
60	1.00000	10.00000	10.00000	0.00000	14.99993	-1.00000
61	1.00000	1.00000	1.00000	0.00000	1.49999	0.00000
62	1.00000	1.00000	1.00000	0.00000	1.49999	1.00000
63	1.00000	1.00000	1.00000	0.00000	1.49999	2.00000
64	1.00000	1.00000	1.00000	0.00000	1.49999	3.00000
65	1.00000	1.00000	1.00000	0.00000	1.49999	4.00000
66	1.00000	1.00000	1.00000	0.00000	1.49999	5.00000
67	1.00000	1.00000	1.00000	0.00000	1.49999	6.00000
68	1.00000	1.00000	1.00000	0.00000	1.49999	7.00000
69	1.00000	1.00000	1.00000	0.00000	1.49999	8.00000
70	1.00000	1.00000	1.00000	0.00000	1.49999	9.00000
71	1.00000	1.00000	1.00000	0.00000	1.49999	10.00000
72	1.00000	1.00000	1.00000	0.00000	1.49999	11.00000
73	1.00000	1.00000	1.00000	0.00000	1.49999	12.00000
74	1.00000	1.00000	1.00000	0.00000	1.49999	13.00000
75	1.00000	1.00000	1.00000	0.00000	1.49999	14.00000
76	1.00000	1.00000	1.00000	0.00000	1.49999	15.00000
77	1.00000	1.00000	1.00000	0.00000	1.49999	16.00000
78	1.00000	1.00000	1.00000	0.00000	1.49999	17.00000
79	1.00000	1.00000	1.00000	0.00000	1.49999	18.00000
80	1.00000	1.00000	1.00000	0.00000	1.49999	19.00000
81	1.00000	1.00000	1.00000	0.00000	1.49999	20.00000
82	1.00000	1.00000	1.00000	0.00000	1.49999	21.00000
83	1.00000	1.00000	1.00000	0.00000	1.49999	22.00000
84	1.00000	1.00000	1.00000	0.00000	1.49999	23.00000
85	1.00000	1.00000	1.00000	0.00000	1.49999	24.00000
86	1.00000	1.00000	1.00000	0.00000	1.49999	25.00000
87	1.00000	1.00000	1.00000	0.00000	1.49999	26.00000
88	1.00000	1.00000	1.00000	0.00000	1.49999	27.00000
89	1.00000	1.00000	1.00000	0.00000	1.49999	28.00000
90	1.00000	1.00000	1.00000	0.00000	1.49999	29.00000
91	1.00000	1.00000	1.00000	0.00000	1.49999	30.00000
92	1.00000	1.00000	1.00000	0.00000	1.49999	31.00000
93	1.00000	1.00000	1.00000	0.00000	1.49999	32.00000
94	1.00000	1.00000	1.00000	0.00000	1.49999	33.00000
95	1.00000	1.00000	1.00000	0.00000	1.49999	34.00000
96	1.00000	1.00000	1.00000	0.00000	1.49999	35.00000
97	1.00000	1.00000	1.00000	0.00000	1.49999	36.00000
98	1.00000	1.00000	1.00000	0.00000	1.49999	37.00000
99	1.00000	1.00000	1.00000	0.00000	1.49999	38.00000
100	1.00000	1.00000	1.00000	0.00000	1.49999	39.00000

Conservation Sums

100.00000	640.00000	0.00000	959.99609	

LCPFCT Test # 3 - Bursting Diaphragm: Step = 100 NX =100 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces
1	1.00000	10.00000	10.00000	0.00000	14.99993	-60.00000
2	1.00000	10.00000	10.00000	0.00000	14.99993	-59.00000
3	1.00000	10.00000	10.00000	0.00000	14.99993	-58.00000
4	1.00000	10.00000	10.00000	0.00000	14.99993	-57.00000
5	1.00000	10.00000	10.00000	0.00000	14.99993	-56.00000
6	1.00000	10.00000	10.00000	0.00000	14.99993	-55.00000
7	1.00000	10.00000	10.00000	0.00000	14.99993	-54.00000
8	1.00000	10.00000	10.00000	0.00000	14.99993	-53.00000
9	1.00000	10.00000	10.00000	0.00000	14.99993	-52.00000
10	1.00000	10.00000	10.00000	0.00000	14.99993	-51.00000

11	1.00000	10.00000	10.00000	0.00000	14.99993	-50.00000
12	1.00000	10.00000	10.00000	0.00000	14.99993	-49.00000
13	1.00000	10.00000	10.00000	0.00000	14.99993	-48.00000
14	1.00000	10.00000	10.00000	0.00000	14.99993	-47.00000
15	1.00000	10.00000	10.00000	0.00000	14.99993	-46.00000
16	1.00000	10.00000	10.00000	0.00000	14.99993	-45.00000
17	1.00000	10.00000	10.00000	0.00000	14.99993	-44.00000
18	1.00000	10.00000	10.00000	0.00000	14.99993	-43.00000
19	1.00000	10.00000	10.00000	0.00000	14.99993	-42.00000
20	1.00000	10.00000	10.00000	0.00000	14.99993	-41.00000
21	1.00000	10.00000	10.00000	0.00000	14.99993	-40.00000
22	1.00000	10.00000	10.00000	0.00000	14.99993	-39.00000
23	1.00000	10.00000	10.00000	0.00000	14.99993	-38.00000
24	1.00000	10.00000	10.00000	0.00000	14.99993	-37.00000
25	1.00000	10.00000	10.00000	0.00000	14.99993	-36.00000
26	1.00000	10.00000	10.00000	0.00000	14.99993	-35.00000
27	1.00000	10.00000	10.00000	0.00000	14.99993	-34.00000
28	1.00000	10.00000	10.00000	0.00000	14.99993	-33.00000
29	1.00000	9.99999	9.99998	0.00000	14.99990	-32.00000
30	1.00000	9.99999	9.99999	0.00001	14.99991	-31.00000
31	0.99999	9.99994	9.99987	0.00001	14.99972	-30.00000
32	0.99998	9.99989	9.99973	0.00007	14.99952	-29.00000
33	0.99997	9.99983	9.99958	0.00025	14.99929	-28.00000
34	0.99980	9.99869	9.99673	0.00023	14.99501	-27.00000
35	0.99982	9.99882	9.99706	0.00228	14.99552	-26.00000
36	0.99848	9.98987	9.97468	0.00208	14.96196	-25.00000
37	0.99862	9.99070	9.97689	0.01654	14.96540	-24.00000
38	0.98965	9.93141	9.82857	0.01511	14.74290	-23.00000
39	0.98994	9.93077	9.83089	0.09434	14.75067	-22.00000
40	0.96048	9.73635	9.35154	0.10614	14.03265	-21.00000
41	0.94729	9.64314	9.13489	0.22795	13.72688	-20.00000
42	0.92858	9.51687	8.83722	0.36219	13.31668	-19.00000
43	0.88410	9.21075	8.14319	0.40569	12.28747	-18.00000
44	0.86856	9.10424	7.90762	0.58334	12.00915	-17.00000
45	0.84777	8.95951	7.59559	0.73331	11.62127	-16.00000
46	0.80170	8.62583	6.91531	0.77048	10.61088	-15.00000
47	0.78551	8.52980	6.70027	0.94461	10.40081	-14.00000
48	0.76956	8.39281	6.45878	1.11127	10.16330	-13.00000
49	0.72772	8.10573	5.89866	1.17294	9.34854	-12.00000
50	0.70716	7.93358	5.61027	1.28551	8.99967	-11.00000
51	0.69716	7.89671	5.50527	1.44648	8.98721	-10.00000
52	0.67841	7.68775	5.21543	1.49895	8.58524	-9.00000
53	0.66366	7.64411	5.07310	1.52963	8.38602	-8.00000
54	0.66272	7.62325	5.05210	1.56000	8.38452	-7.00000
55	0.66338	7.60589	5.04558	1.57740	8.39363	-6.00000
56	0.66483	7.59057	5.04644	1.57395	8.39312	-5.00000
57	0.66562	7.59281	5.05392	1.56228	8.39313	-4.00000
58	0.66652	7.62785	5.08409	1.55166	8.42847	-3.00000
59	0.66669	7.68111	5.12091	1.54293	8.47489	-2.00000
60	0.66513	7.73356	5.14382	1.51090	8.47487	-1.00000
61	0.65147	7.83471	5.10405	1.54063	8.42918	0.00000
62	0.65036	7.81949	5.08548	1.54358	8.40298	1.00000
63	0.65034	7.80526	5.07610	1.54576	8.39107	2.00000
64	0.65080	7.79762	5.07468	1.54735	8.39108	3.00000
65	0.65583	7.78186	5.10358	1.53769	8.43069	4.00000
66	0.68089	7.69643	5.24046	1.48121	8.60758	5.00000
67	0.92725	5.31404	4.92742	1.63829	8.63545	6.00000
68	1.41717	3.68566	5.22322	1.48923	9.40629	7.00000
69	1.74176	2.89699	5.04585	1.54815	9.65604	8.00000
70	2.03016	2.58020	5.23824	1.57113	10.36299	9.00000
71	2.17146	2.38683	5.18292	1.56674	10.43944	10.00000

72	2.20853	2.34843	5.18657	1.55193	10.43942	11.00000
73	2.25440	2.31318	5.21484	1.52026	10.42741	12.00000
74	2.25398	1.05637	2.38104	1.18232	5.14694	13.00000
75	1.15678	0.86613	1.00192	0.00195	1.50288	14.00000
76	1.00002	0.99998	1.00000	0.00000	1.49999	15.00000
77	1.00000	1.00000	1.00000	0.00000	1.49999	16.00000
78	1.00000	1.00000	1.00000	0.00000	1.49999	17.00000
79	1.00000	1.00000	1.00000	0.00000	1.49999	18.00000
80	1.00000	1.00000	1.00000	0.00000	1.49999	19.00000
81	1.00000	1.00000	1.00000	0.00000	1.49999	20.00000
82	1.00000	1.00000	1.00000	0.00000	1.49999	21.00000
83	1.00000	1.00000	1.00000	0.00000	1.49999	22.00000
84	1.00000	1.00000	1.00000	0.00000	1.49999	23.00000
85	1.00000	1.00000	1.00000	0.00000	1.49999	24.00000
86	1.00000	1.00000	1.00000	0.00000	1.49999	25.00000
87	1.00000	1.00000	1.00000	0.00000	1.49999	26.00000
88	1.00000	1.00000	1.00000	0.00000	1.49999	27.00000
89	1.00000	1.00000	1.00000	0.00000	1.49999	28.00000
90	1.00000	1.00000	1.00000	0.00000	1.49999	29.00000
91	1.00000	1.00000	1.00000	0.00000	1.49999	30.00000
92	1.00000	1.00000	1.00000	0.00000	1.49999	31.00000
93	1.00000	1.00000	1.00000	0.00000	1.49999	32.00000
94	1.00000	1.00000	1.00000	0.00000	1.49999	33.00000
95	1.00000	1.00000	1.00000	0.00000	1.49999	34.00000
96	1.00000	1.00000	1.00000	0.00000	1.49999	35.00000
97	1.00000	1.00000	1.00000	0.00000	1.49999	36.00000
98	1.00000	1.00000	1.00000	0.00000	1.49999	37.00000
99	1.00000	1.00000	1.00000	0.00000	1.49999	38.00000
100	1.00000	1.00000	1.00000	0.00000	1.49999	39.00000

Conservation Sums

100.00002	618.58447	45.00003	959.99567	

LCPFCT Test # 3 - Bursting Diaphragm: Step = 200 NX =100 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces
1	1.00000	10.00000	10.00000	0.00000	14.99993	-60.00000
2	1.00000	10.00000	10.00000	0.00000	14.99993	-59.00000
3	1.00000	10.00000	10.00000	0.00000	14.99993	-58.12500
4	1.00000	10.00000	10.00000	0.00000	14.99993	-57.12500
5	1.00000	10.00000	10.00000	0.00000	14.99993	-56.12500
6	1.00000	10.00000	10.00000	0.00000	14.99993	-55.12500
7	1.00000	10.00000	10.00000	0.00000	14.99992	-54.12500
8	1.00000	10.00000	10.00000	0.00000	14.99992	-53.12500
9	1.00000	9.99998	9.99995	0.00000	14.99985	-52.12500
10	1.00000	9.99998	9.99996	0.00003	14.99986	-51.12500
11	0.99998	9.99985	9.99962	0.00003	14.99936	-50.12500
12	0.99998	9.99985	9.99962	0.00022	14.99936	-49.12500
13	0.99990	9.99934	9.99835	0.00034	14.99746	-48.12500
14	0.99971	9.99804	9.99512	0.00065	14.99261	-47.12500
15	0.99968	9.99785	9.99464	0.00369	14.99189	-46.12500
16	0.99760	9.98400	9.96009	0.00406	14.94006	-45.12500
17	0.99689	9.97919	9.94812	0.02219	14.92234	-44.12500
18	0.98951	9.92987	9.82568	0.03272	14.73898	-43.12500
19	0.98341	9.88910	9.72506	0.07242	14.59009	-42.12500
20	0.97335	9.82158	9.55989	0.11805	14.34654	-41.12500
21	0.95750	9.71456	9.30166	0.15631	13.96412	-40.12500
22	0.94812	9.65114	9.15047	0.23294	13.75137	-39.12500
23	0.93204	9.54183	8.89338	0.28739	13.37850	-38.12500
24	0.91363	9.41560	8.60241	0.33418	12.95457	-37.12500

25	0.90596	9.36289	8.48237	0.42965	12.80711	-36.12500
26	0.88679	9.23072	8.18576	0.47745	12.37965	-35.12500
27	0.86797	9.09940	7.89798	0.52698	11.96743	-34.12500
28	0.86217	9.05890	7.81035	0.63439	11.88896	-33.12500
29	0.84276	8.92297	7.51994	0.67994	11.47467	-32.12500
30	0.82269	8.78013	7.22329	0.71973	11.04796	-31.12500
31	0.81802	8.74870	7.15666	0.83227	11.01824	-30.12500
32	0.80185	8.63083	6.92063	0.89444	10.70164	-29.12500
33	0.77902	8.46884	6.59740	0.92035	10.22599	-28.12500
34	0.77111	8.41151	6.48617	1.02204	10.13195	-27.12500
35	0.76143	8.34473	6.35395	1.11207	10.00170	-26.12500
36	0.73882	8.16852	6.03510	1.14566	9.53747	-25.12500
37	0.72447	8.07598	5.85077	1.20772	9.30446	-24.12500
38	0.71897	8.02844	5.77222	1.31747	9.28225	-23.12500
39	0.70355	7.90964	5.56482	1.37106	9.00846	-22.12500
40	0.68684	7.78385	5.34622	1.40380	8.69604	-21.12500
41	0.68161	7.75109	5.28326	1.46901	8.66030	-20.12500
42	0.67739	7.71643	5.22707	1.51275	8.61565	-19.12500
43	0.67053	7.65705	5.13427	1.52826	8.48440	-18.12500
44	0.66758	7.64227	5.10184	1.53396	8.43814	-17.12500
45	0.66684	7.63888	5.09395	1.54507	8.43684	-16.12500
46	0.66625	7.63422	5.08632	1.55192	8.43176	-15.12500
47	0.66625	7.63087	5.08408	1.55164	8.42812	-14.12500
48	0.66627	7.63346	5.08594	1.54856	8.42773	-13.12500
49	0.66628	7.63585	5.08764	1.54607	8.42774	-12.12500
50	0.66724	7.64135	5.09858	1.54376	8.44291	-11.12500
51	0.66742	7.64384	5.10165	1.54263	8.44658	-10.12500
52	0.66742	7.64411	5.10183	1.54237	8.44658	-9.12500
53	0.66737	7.64329	5.10090	1.54249	8.44524	-8.12500
54	0.66720	7.64180	5.09862	1.54357	8.44274	-7.12500
55	0.66707	7.64217	5.09789	1.54417	8.44211	-6.12500
56	0.66707	7.64237	5.09802	1.54417	8.44231	-5.12500
57	0.66708	7.64340	5.09879	1.54312	8.44238	-4.12500
58	0.66752	7.65289	5.10846	1.54120	8.45543	-3.12500
59	0.66868	7.65532	5.11893	1.53476	8.46588	-2.12500
60	0.66890	7.65729	5.12197	1.53394	8.46988	-1.12500
61	0.66897	7.65664	5.12203	1.53380	8.46988	-0.12500
62	0.66897	7.65388	5.12018	1.53648	8.46987	0.87500
63	0.66833	7.63431	5.10222	1.53785	8.44358	1.87500
64	0.66651	7.63895	5.09142	1.55425	8.44213	2.87500
65	0.66625	7.58955	5.05655	1.55738	8.39276	3.87500
66	0.66370	7.61252	5.05244	1.56335	8.38968	4.87500
67	0.65725	7.70666	5.06517	1.55243	8.38971	5.87500
68	0.65723	7.76724	5.10488	1.55149	8.44830	6.87500
69	0.65542	7.82882	5.13116	1.53026	8.46410	7.87500
70	0.65385	7.84153	5.12715	1.52963	8.45562	8.87500
71	0.65385	7.83928	5.12568	1.52963	8.45341	9.87500
72	0.65474	7.82925	5.12613	1.53018	8.45568	10.87500
73	0.65613	7.85534	5.15414	1.53019	8.49933	11.87500
74	0.78982	6.55305	5.17573	1.51305	8.66764	12.87500
75	1.03607	4.65581	4.82377	1.65827	8.66014	13.87500
76	1.54017	3.45870	5.32697	1.46551	9.64434	14.87500
77	1.77799	2.80279	4.98333	1.54795	9.60513	15.87500
78	2.03304	2.49940	5.08139	1.52444	9.98437	16.87500
79	2.17834	2.35766	5.13577	1.55075	10.32288	17.87500
80	2.28334	2.25747	5.15459	1.55524	10.49328	18.87500
81	2.34451	2.18807	5.12996	1.54653	10.49866	19.87500
82	2.38527	2.14067	5.10608	1.54237	10.49624	20.87500
83	2.41726	2.10460	5.08736	1.54032	10.49859	21.87500
84	2.43101	2.09506	5.09311	1.53851	10.51675	22.87500
85	2.43224	2.09415	5.09348	1.53812	10.51730	23.87500

86	2.43350	2.09330	5.09403	1.53723	10.51626	24.87500
87	2.43284	1.76175	4.28607	1.39070	8.78168	25.87500
88	1.13769	0.98511	1.12075	0.08531	1.68525	26.87500
89	1.00005	1.00002	1.00007	0.00006	1.50009	27.87500
90	1.00000	1.00000	1.00000	0.00000	1.49999	28.87500
91	1.00000	1.00000	1.00000	0.00000	1.49999	29.87500
92	1.00000	1.00000	1.00000	0.00000	1.49999	30.87500
93	1.00000	1.00000	1.00000	0.00000	1.49999	31.87500
94	1.00000	1.00000	1.00000	0.00000	1.49999	32.87500
95	1.00000	1.00000	1.00000	0.00000	1.49999	33.87500
96	1.00000	1.00000	1.00000	0.00000	1.49999	34.87500
97	1.00000	1.00000	1.00000	0.00000	1.49999	35.87500
98	1.00000	1.00000	1.00000	0.00000	1.49999	36.87500
99	1.00000	1.00000	1.00000	0.00000	1.49999	38.00000
100	1.00000	1.00000	1.00000	0.00000	1.49999	39.00000

Conservation Sums

100.00005	597.52051	90.34889	961.68964	

LCPFCT Test # 3 - Bursting Diaphragm: Step =1600 NX =100 DT = 0.050

I	Density	Temperature	Pressure	Velocity	Energy	Interfaces
1	0.99906	9.99370	9.98426	0.00002	14.97631	-60.00000
2	0.99906	9.99370	9.98426	0.00003	14.97631	-59.00000
3	0.99905	9.99367	9.98419	0.00006	14.97621	-58.12500
4	0.99900	9.99332	9.98330	0.00029	14.97488	-57.12500
5	0.99878	9.99189	9.97973	0.00118	14.96952	-56.12500
6	0.99824	9.98827	9.97068	0.00342	14.95596	-55.12500
7	0.99679	9.97859	9.94652	0.00937	14.91974	-54.12500
8	0.99297	9.95316	9.88321	0.02499	14.82505	-53.12500
9	0.98515	9.90085	9.75378	0.05718	14.63221	-52.12500
10	0.97309	9.82000	9.55575	0.10709	14.33913	-51.12500
11	0.95842	9.72114	9.31697	0.16837	13.98897	-50.12500
12	0.94323	9.61813	9.07207	0.23254	13.63353	-49.12500
13	0.92872	9.51927	8.84071	0.29445	13.30126	-48.12500
14	0.91508	9.42587	8.62538	0.35325	12.99511	-47.12500
15	0.90196	9.33563	8.42033	0.41035	12.70638	-46.12500
16	0.88900	9.24609	8.21978	0.46727	12.42667	-45.12500
17	0.87605	9.15614	8.02122	0.52474	12.15238	-44.12500
18	0.86311	9.06586	7.82484	0.58271	11.88373	-43.12500
19	0.85026	8.97572	7.63169	0.64088	11.62208	-42.12500
20	0.83754	8.88610	7.44247	0.69900	11.36827	-41.12500
21	0.82497	8.79707	7.25734	0.75703	11.12235	-40.12500
22	0.81254	8.70859	7.07610	0.81501	10.88396	-39.12500
23	0.80024	8.62059	6.89854	0.87296	10.65268	-38.12500
24	0.78806	8.53303	6.72458	0.93092	10.42830	-37.12500
25	0.77602	8.44599	6.55425	0.98883	10.21072	-36.12500
26	0.76411	8.35951	6.38759	1.04668	9.99990	-35.12500
27	0.75235	8.27367	6.22472	1.10439	9.79584	-34.12500
28	0.74076	8.18860	6.06578	1.16189	9.59863	-33.12500
29	0.72936	8.10454	5.91116	1.21899	9.40859	-32.12500
30	0.71822	8.02194	5.76155	1.27539	9.22641	-31.12500
31	0.70743	7.94155	5.61813	1.33057	9.05338	-30.12500
32	0.69715	7.86451	5.48275	1.38369	8.89147	-29.12500
33	0.68762	7.79281	5.35853	1.43338	8.74414	-28.12500
34	0.67931	7.72987	5.25095	1.47715	8.61751	-27.12500
35	0.67292	7.68146	5.16899	1.51101	8.52163	-26.12500
36	0.66923	7.65318	5.12172	1.53073	8.46659	-25.12500
37	0.66810	7.64466	5.10738	1.53670	8.44987	-24.12500
38	0.66805	7.64429	5.10673	1.53703	8.44917	-23.12500

39	0.66805	7.64435	5.10683	1.53701	8.44931	-22.12500
40	0.66805	7.64437	5.10685	1.53701	8.44934	-21.12500
41	0.66805	7.64436	5.10684	1.53704	8.44936	-20.12500
42	0.66805	7.64433	5.10683	1.53706	8.44936	-19.12500
43	0.66804	7.64422	5.10664	1.53715	8.44915	-18.12500
44	0.66794	7.64347	5.10538	1.53771	8.44773	-17.12500
45	0.66783	7.64261	5.10395	1.53833	8.44609	-16.12500
46	0.66782	7.64251	5.10379	1.53840	8.44590	-15.12500
47	0.66782	7.64256	5.10383	1.53834	8.44590	-14.12500
48	0.66798	7.64378	5.10588	1.53762	8.44842	-13.12500
49	0.66813	7.64483	5.10772	1.53685	8.45056	-12.12500
50	0.66814	7.64488	5.10783	1.53677	8.45066	-11.12500
51	0.66811	7.64466	5.10749	1.53685	8.45021	-10.12500
52	0.66793	7.64356	5.10540	1.53781	8.44784	-9.12500
53	0.66791	7.64341	5.10511	1.53799	8.44758	-8.12500
54	0.66791	7.64350	5.10519	1.53799	8.44768	-7.12500
55	0.66811	7.64445	5.10737	1.53701	8.45019	-6.12500
56	0.66856	7.64806	5.11315	1.53465	8.45696	-5.12500
57	0.66861	7.64857	5.11391	1.53437	8.45789	-4.12500
58	0.66861	7.64852	5.11387	1.53441	8.45787	-3.12500
59	0.66845	7.64767	5.11209	1.53507	8.45568	-2.12500
60	0.66775	7.64231	5.10317	1.53904	8.44555	-1.12500
61	0.66697	7.63552	5.09266	1.54296	8.43289	-0.12500
62	0.66678	7.63463	5.09065	1.54424	8.43097	0.87500
63	0.66678	7.63458	5.09062	1.54438	8.43107	1.87500
64	0.66680	7.63467	5.09077	1.54435	8.43128	2.87500
65	0.66703	7.63669	5.09391	1.54308	8.43496	3.87500
66	0.66769	7.64179	5.10233	1.53938	8.44457	4.87500
67	0.66857	7.64834	5.11342	1.53482	8.45755	5.87500
68	0.66913	7.65279	5.12070	1.53144	8.46567	6.87500
69	0.66932	7.65369	5.12280	1.53080	8.46839	7.87500
70	0.66935	7.65379	5.12304	1.53082	8.46879	8.87500
71	0.66935	7.65373	5.12300	1.53088	8.46879	9.87500
72	0.66924	7.65305	5.12169	1.53149	8.46733	10.87500
73	0.66890	7.64983	5.11700	1.53349	8.46196	11.87500
74	0.66838	7.64725	5.11129	1.53616	8.45551	12.87500
75	0.66738	7.64728	5.10365	1.53833	8.44510	13.87500
76	0.66728	7.64508	5.10144	1.54187	8.44531	14.87500
77	0.66728	7.62649	5.08904	1.54161	8.42644	15.87500
78	0.66843	7.60479	5.08329	1.54862	8.42643	16.87500
79	0.67474	7.53155	5.08187	1.54372	8.42675	17.87500
80	1.09444	4.61876	5.05496	1.55191	8.90033	18.87500
81	2.00489	2.58898	5.19062	1.52310	10.11138	19.87500
82	2.32824	2.19679	5.11465	1.53662	10.42066	20.87500
83	2.34057	2.18357	5.11081	1.53809	10.43474	21.87500
84	2.34058	2.18359	5.11085	1.53811	10.43488	22.87500
85	2.34057	2.18359	5.11084	1.53810	10.43485	23.87500
86	2.34057	2.18358	5.11084	1.53812	10.43491	24.87500
87	2.34131	2.18284	5.11070	1.53860	10.43731	25.87500
88	2.34447	2.17930	5.10929	1.53847	10.43843	26.87500
89	2.34484	2.17887	5.10910	1.53834	10.43814	27.87500
90	2.34498	2.17824	5.10793	1.53878	10.43814	28.87500
91	2.34752	2.17635	5.10903	1.53940	10.44501	29.87500
92	2.34828	2.17718	5.11263	1.53984	10.45292	30.87500
93	2.34827	2.17718	5.11261	1.53983	10.45286	31.87500
94	2.34412	2.17607	5.10096	1.53640	10.41807	32.87500
95	1.89018	1.92088	3.63081	1.15993	6.71776	33.87500
96	1.05977	1.11000	1.17635	0.11801	1.77189	34.87500
97	1.00002	1.00002	1.00004	0.00003	1.50006	35.87500
98	1.00000	1.00000	1.00000	0.00000	1.49999	36.87500
99	1.00000	1.00000	1.00000	0.00000	1.49999	38.00000

```
100      1.00000      1.00000      1.00000      0.00000      1.49999     39.00000

Conservation Sums
       99.96184                 585.24878    116.32616    962.41534
```

```
C=====================================================================

          Program FAST2D

C---------------------------------------------------------------------
c
c  BURSTING DIAPHRAGM "MUZZLE FLASH" - LCPFCT TEST # 4       August 1992
c
c  The problem begins with 1000:1 pressure and 100:1 density ratios
c  across a diaphragm inside a solid cylindrical barrel.  The ideal wall
c  of the barrel is 10 cells thick (1.0 cm) with its inner radius given
c  as 1.5 cm and its outer radius of 2.5 cm.  The run starts at time
c  t = 0.0 when the diaphragm at interface J = 11 (inside the barrel) is
c  ruptured.  The flow then expands upward in a 1D manner, spilling out
c  of the barrel in a 2D flow which eventually reaches the boundaries at
c  R = 4.0 cm and Z = 4.0 cm where a very simple extrapolative outflow
c  condition is expressed through the LCPFCT boundary conditions values.
c  The outflow condition used here includes a slow relaxation to ambient
c  conditions far from the origin.
c
C---------------------------------------------------------------------

         Implicit  NONE

         Integer   NPT, I, J, IJ
         Parameter ( NPT = 202 )
         Integer   NR, NRP, MR, IALFR,     BC_AXIS, BC_WALL, BC_OUTF
         Integer   NZ, NZP, MZ, IALFZ,     LOUT, MAXSTP, IPRINT
         Integer   ICIN, ICOUT, JCTOP,     JSTEP, ISTEP
         Real      DR,          DZ,        DT,          TIME
         Real      COURANT,     DTNEW,     VTYPICAL,    RELAX
         Real      DTMAX,       VZMAX,     R(NPT),      Z(NPT)
         Real      RHO(40,40), RVR(40,40), RVZ(40,40), ERG(40,40)

         Real      RHO_IN,      PRE_IN,    VEL_IN,      GAMMA0
         Real      RHOAMB,      PREAMB,    VELAMB,      GAMMAM
         Real      RHON(NPT),   RVRN(NPT),  RVTN(NPT),   ERGN(NPT)
         Common    / ARRAYS / RHON,   RVRN,   RVTN,   ERGN,   RELAX,
     &                        RHO_IN, PRE_IN, VEL_IN, GAMMA0,
     &                        RHOAMB, PREAMB, VELAMB, GAMMAM

 1000 Format ('1', /, '   LCPFCT Test # 4 - FAST2D Barrel Explosion:',
     1        '  Step =', I4, /, 5X, I3, ' x', I3 ' Uniform Grid.',
     2        '  Time=', 1PE12.4, ' and   DT =', E12.4 )
 1005 Format (' After step ', I5, '  TIME = ', 1PE12.4,
     &           ' and timestep DT = ', E12.4 )
 1010 Format (1X, /, ' Fluid variables on selected lines ',
     1          /, ' I, J', '    RHO axis VZ     PRE ',
     2                      '    RHO top  VR     PRE ',
     3                      '    RHO wall VZ     PRE ', / )
 1011 Format ( I3, 1X, 3(1X, F8.2, F8.2, F8.2) )

c  The 2D barrel explosion program control parameters are specified.
c  (Change here to run other cases) . . .
C---------------------------------------------------------------------
      NR      =  40  ! Number of cells in the first (radial R) direction
      IALFR   =   2  ! Sets cylindrical coordinates in the R direction
      DR      =  0.1 ! Cell size (e.g., cm) in the radial direction
      NZ      =  40  ! Number of cells in the second (axial Z) direction
      IALFZ   =   1  ! Sets Cartesian coordinates in the Z direction
      DZ      =  0.1 ! Cell size (e.g., cm) in the axial direction
```

```
      LOUT    =  11  ! Logical unit number of printed output device
      BC_AXIS =   1  ! Cylindrical axis set as an impermeable wall
      BC_OUTF =   2  ! Outer boundaries set as extrapolative outflow
      BC_WALL =   1  ! Walls of the barrel set as an ideal solid wall
      MAXSTP  = 401  ! Maximum number of timesteps of length DT
      IPRINT  =  25  ! Initial frequency of validation printout results
      COURANT =  0.4 ! Approximate maximum Cournat number allowed
      DTMAX = 2.0E-7 ! Maximum timestep allowed in the computation
      DT    = 1.0E-9 ! Initial (small guess) for starting timestep

c  Initialize the test problem geometry, a cylindrical shell JCTOP cells
c  high in Z (indexed by J) which extends from the left of cell ICIN to
c  the right of cell ICOUT in X (indexed by I).
C-------------------------------------------------------------------------
      ICIN    =  16  ! Number of the innermost radial cell in the barrel
      ICOUT   =  25  ! Number of the outermost radial cell in the barrel
      JCTOP   =  20  ! Number of the uppermost axial cell in the barrel
      GAMMA0 = 1.4   ! Gas constant
      RHOAMB = 0.00129       ! Initialization and relaxation BCN = 2
      PREAMB = 1.013E+6      ! Initialization and relaxation BCN = 2
      VELAMB = 0.0           ! Initialization and relaxation BCN = 2
      RHO_IN = 100.0*RHOAMB  ! Initialization and relaxation BC1 = 2
      PRE_IN = 1000.0*PREAMB ! Initialization and relaxation BC1 = 2
      VEL_IN = 0.0           ! Initialization and relaxation BC1 = 2
      RELAX =  0.002 ! Relaxation rate, used when BC1 or BCN = 2
      GAMMAM = GAMMA0 - 1.0

c  Determine the cell interface locations, here a uniform grid . . .
C-------------------------------------------------------------------------
      NRP = NR + 1
      NZP = NZ + 1
      Do 100 I = 1, NRP
  100    R(I) = DR*FLOAT(I-1)
      Do 110 J = 1, NZP
  110    Z(J) = DZ*FLOAT(J-1)

c  Fill the arrays with air at STP and behind the diaphragm increase the
c  density by 100 to 1 and the pressure by 1000 to 1 . . .
C-------------------------------------------------------------------------
      Do 200 J = 1, NZ
      Do 200 I = 1, NR
         RHO(I,J) = RHOAMB
         RVR(I,J) = 0.0
         RVZ(I,J) = 0.0
  200    ERG(I,J) = PREAMB/GAMMAM
      Do 210 J = 1, JCTOP/2
      Do 210 I = 1, ICIN - 1
         ERG(I,J) = PRE_IN/GAMMAM
  210    RHO(I,J) = RHO_IN

c  Mark the unused cells inside the cylindrical 'barrel' so they will
c  show up distinctly compared to ambient values in the plots.  This
c  has no effect as the simulation does not access these values . . .
C-------------------------------------------------------------------------
      Do 220 J = 1, JCTOP
      Do 220 I = ICIN, ICOUT
         ERG(I,J) = 20.0*ERG(I,J)
  220    RHO(I,J) = 20.0*RHO(I,J)

c  Begin loop over the timesteps . . .
C-------------------------------------------------------------------------
```

```
      TIME = 0.0
      Do 9999 ISTEP = 1, MAXSTP

c  Compute the next timestep based on a 'Courant' number COURANT . . .
C-----------------------------------------------------------------------
         VZMAX = 0.0
         Do 240 J = 1, NZ
         Do 240 I = 1, NR
            VTYPICAL = ERG(I,J)/RHO(I,J)
  240       VZMAX = AMAX1 ( VTYPICAL, VZMAX )
         VZMAX = SQRT ( VZMAX )
         DTNEW = COURANT*AMIN1(DR,DZ)/VZMAX
         DT = AMIN1 ( DTMAX, 1.25*DT, DTNEW )

c  The results are printed when required . . .
C-----------------------------------------------------------------------
         JSTEP = ISTEP - 1
         If ( MOD(JSTEP,5) .eq. 0 ) Write ( 6, 1005 ) JSTEP, TIME, DT
         If ( MOD(JSTEP,IPRINT) .eq. 0 ) Then
            Write ( LOUT, 1000 ) ISTEP, NR, NZ, TIME, DT
            If ( ISTEP .ge. 4*IPRINT ) IPRINT = 2*IPRINT
            Write ( LOUT, 1010 )
            Do 230 IJ = 1, 40
               DIN(1) = RHO(1,IJ)/RHOAMB
               DIN(2) = 0.01*RVZ(1,IJ)/RHO(1,IJ)
               DIN(3) = (GAMMAM/PREAMB)*(ERG(1,IJ) - 0.5*
     &                   (RVR(1,IJ)**2 + RVZ(1,IJ)**2)/RHO(1,IJ))
               DIN(4) = RHO(IJ,40)/RHOAMB
               DIN(5) = 0.01*RVR(IJ,40)/RHO(IJ,40)
               DIN(6) = (GAMMAM/PREAMB)*(ERG(IJ,40) - 0.5*
     &                   (RVR(IJ,40)**2 + RVZ(IJ,40)**2)/RHO(IJ,40))
               DIN(7) = RHO(40,IJ)/RHOAMB
               DIN(8) = 0.01*RVZ(40,IJ)/RHO(40,IJ)
               DIN(9) = (GAMMAM/PREAMB)*(ERG(40,IJ) - 0.5*
     &                   (RVR(40,IJ)**2 + RVZ(40,IJ)**2)/RHO(40,IJ))
  230          Write ( LOUT, 1011 ) IJ, ( DIN(I), I = 1, 9 )
            Write ( LOUT, 1011 )

c  Integrate the fluid equations in the radial direction (indexed by I).
c  The outer boundary condition at interface I = NR+1 is an extra-
c  polation from the interior cell values with a slow relaxation to
c  the known distant ambient conditions . . .
C-----------------------------------------------------------------------
         Call RESIDIFF ( 0.999 )
         Call MAKEGRID ( R, R, 1, NRP, IALFR )
         Do 300 J = 1, NZ

c  Pick up the data from the 2D arrays in the radial direction, setting
c  the temporary, compact 1D arrays for GASDYN . . .
C-----------------------------------------------------------------------
            Do 400 I = 1, NR
               RHON(I) = RHO(I,J)
               RVRN(I) = RVR(I,J)
               RVTN(I) = RVZ(I,J)
  400          ERGN(I) = ERG(I,J)

c  Integrate along the radials inside and outside the cylinder  . . .
C-----------------------------------------------------------------------
            If ( J .le. JCTOP ) Then
               Call GASDYN ( 1, ICIN-1, BC_AXIS, BC_WALL, DT )
               Call GASDYN ( ICOUT+1, NR, BC_WALL, BC_OUTF, DT)
```

```
c  Integrate along the radials (indexing in I) above the cylinder
c  which reach from the axis to the outer boundary . ..
C-----------------------------------------------------------------------
            Else
               Call GASDYN ( 1, NR, BC_AXIS, BC_OUTF, DT )
            End If

c  Put the data back into the 2D arrays in the radial direction . . .
C-----------------------------------------------------------------------
            Do 500 I = 1, NR
               RHO(I,J) = RHON(I)
               RVR(I,J) = RVRN(I)
               RVZ(I,J) = RVTN(I)
  500          ERG(I,J) = ERGN(I)
  300    Continue     ! End loop integrating the NZ rows.

c  Integrate along the axials (indexing in J) which reach from the
c  lower active J cell (1 or JCTOP+1) to the upper boundary.  The
c  upper boundary condition at interface J = NZ+1 (BCN = 2 ) is an
c  extrapolation from the interior cell values with a slow relaxation
c  to the known distant ambient conditions . . .
C-----------------------------------------------------------------------
         Call MAKEGRID ( Z, Z, 1, NZP, IALFZ )
         Do 600 I = 1, NR

c  Pick up the data from the 2D arrays in the axial direction, setting
c  the temporary, compact 1D arrays for GASDYN . . .
C-----------------------------------------------------------------------
            Do 700 J = 1, NZ
               RHON(J) = RHO(I,J)
               RVTN(J) = RVR(I,J)
               RVRN(J) = RVZ(I,J)
  700          ERGN(J) = ERG(I,J)

c  Integrate along the axials either from the lower solid boundary at
c  interface J = 1 or from the top of the barrel at J = 21 for cells
c  with I = ICIN to ICOUT . . .
C-----------------------------------------------------------------------
            If ( I.ge.ICIN .and. I.le.ICOUT ) Then
               Call GASDYN ( JCTOP+1, NZ, BC_WALL, BC_OUTF, DT)
            Else
               Call GASDYN ( 1, NZ, BC_WALL, BC_OUTF, DT )
            End If

c  Put the data back into the 2D arrays in the axial direction . . .
C-----------------------------------------------------------------------
            Do 800 J = 1, NZ
               RHO(I,J) = RHON(J)
               RVR(I,J) = RVTN(J)
               RVZ(I,J) = RVRN(J)
  800          ERG(I,J) = ERGN(J)
  600    Continue     ! End loop integrating the NR columns.

         TIME = TIME + DT

 9999 Continue        ! End of the timestep loop.

      Stop
      End
```

```
C=====================================================================
  LCPFCT Test # 4 - FAST2D Barrel Explosion:  Step =   1
     40 x 40 Uniform Grid.  Time=  0.0000E+00 and   DT =  1.2500E-09

Fluid variables on selected lines
I, J  RHO axis VZ     PRE      RHO top  VR     PRE    RHO wall VZ     PRE

 1   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 2   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 3   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 4   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 5   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 6   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 7   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 8   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
 9   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
10   100.00    0.00 1000.00     1.00    0.00    1.00   1.00    0.00    1.00
11     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
12     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
13     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
14     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
15     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
16     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
17     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
18     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
19     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
20     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
21     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
22     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
23     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
24     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
25     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
26     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
27     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
28     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
29     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
30     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
31     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
32     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
33     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
34     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
35     1.00    0.00    1.00     1.00    0.00    1.00   1.00    0.00    1.00
```

36	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00
37	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00
38	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00
39	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00
40	1.00	0.00	1.00	1.00	0.00	1.00	1.00	0.00	1.00

LCPFCT Test # 4 - FAST2D Barrel Explosion: Step = 151
40 x 40 Uniform Grid. Time= 2.5060E-05 and DT = 2.0000E-07

Fluid variables on selected lines

I, J	RHO axis	VZ	PRE	RHO top	VR	PRE	RHO wall	VZ	PRE
1	25.19	30.06	146.86	12.01	0.01	53.98	1.00	0.00	1.00
2	25.28	71.50	147.34	11.99	0.05	53.87	1.00	0.00	1.00
3	25.37	119.77	147.30	11.97	7.40	53.76	1.00	0.00	1.00
4	25.38	165.83	147.59	11.63	10.37	51.75	1.00	0.00	1.00
5	25.36	213.07	146.95	11.41	43.71	50.52	1.00	0.00	1.00
6	25.22	264.63	146.30	10.56	63.09	45.72	1.00	0.00	1.00
7	25.13	311.24	144.66	9.95	114.77	41.94	1.00	0.00	1.00
8	25.10	362.53	144.01	9.36	164.56	38.70	1.00	0.00	1.00
9	25.16	420.76	144.98	8.38	196.37	32.85	1.00	0.00	1.00
10	25.17	467.50	143.35	7.94	258.82	30.28	1.00	0.00	1.00
11	24.95	512.41	140.63	6.95	307.26	25.98	1.00	0.00	1.00
12	24.80	566.81	138.89	6.55	359.17	22.79	1.00	0.00	1.00
13	24.57	620.80	138.12	5.80	410.66	19.67	1.00	0.00	1.00
14	24.09	672.04	136.21	5.28	459.34	17.19	1.00	0.00	1.00
15	23.67	723.80	134.31	4.79	513.13	15.18	1.00	0.00	1.00
16	23.32	773.53	132.10	4.13	597.67	12.89	1.00	0.00	1.00
17	22.94	820.67	129.48	3.57	691.17	9.44	1.00	0.00	1.00
18	22.60	869.17	127.29	2.89	747.93	6.87	1.00	0.00	1.00
19	22.30	918.10	125.13	2.40	818.20	6.47	1.00	0.00	1.00
20	22.14	968.59	123.10	2.14	918.84	5.24	1.00	0.00	1.00
21	22.05	1019.32	121.19	1.77	992.51	3.48	1.00	0.00	1.00
22	21.82	1068.43	118.99	1.39	1033.27	3.07	1.00	0.00	1.00
23	21.54	1117.82	116.80	1.18	1171.59	2.68	1.00	0.00	1.00
24	21.13	1165.29	114.09	0.86	1285.47	1.71	1.00	0.00	1.00
25	20.63	1213.81	111.28	0.75	1279.33	1.46	1.00	0.00	1.00
26	20.21	1262.16	108.56	0.75	1279.10	1.27	1.00	0.00	1.00
27	19.73	1307.89	105.37	1.16	990.03	3.36	1.00	0.00	1.00
28	19.18	1355.07	101.86	1.72	772.98	5.96	1.00	0.00	1.00
29	18.66	1398.05	98.70	1.78	748.22	5.97	1.00	0.00	1.00
30	18.08	1443.14	95.42	1.93	717.76	8.02	1.00	0.00	1.00

31	17.48	1489.85	89.78	2.41	767.94	8.42	1.00	0.00	1.00
32	16.95	1532.89	86.12	2.73	722.70	9.10	1.00	0.00	1.00
33	16.54	1573.44	83.37	2.77	711.28	9.21	1.00	0.00	1.00
34	16.15	1614.41	80.25	2.77	594.68	4.72	1.00	0.00	1.00
35	15.71	1659.32	73.69	1.11	0.55	1.00	1.00	0.00	1.00
36	14.75	1689.00	65.28	1.00	0.00	1.00	1.00	0.00	1.00
37	13.73	1699.93	63.11	1.00	0.00	1.00	1.00	0.00	1.00
38	13.44	1724.09	61.23	1.00	0.00	1.00	1.00	0.00	1.00
39	12.58	1785.64	56.42	1.00	0.00	1.00	1.00	0.00	1.00
40	12.01	1800.57	53.98	1.00	0.00	1.00	1.00	0.00	1.00

LCPFCT Test # 4 - FAST2D Barrel Explosion: Step = 401
40 x 40 Uniform Grid. Time= 7.5061E-05 and DT = 2.0000E-07

Fluid variables on selected lines

I, J	RHO	axis VZ	PRE	RHO	top VR	PRE	RHO	wall VZ	PRE
1	6.72	10.05	23.15	2.15	2.07	4.60	1.03	1.65	1.05
2	6.72	22.34	23.15	2.14	40.89	4.56	1.05	-18.54	1.06
3	6.72	40.14	23.11	2.12	61.83	4.51	1.24	-67.51	1.36
4	6.71	55.56	23.03	2.09	97.61	4.42	1.39	-101.13	1.59
5	6.68	74.31	22.91	2.05	114.92	4.35	1.42	-116.58	1.64
6	6.65	89.89	22.71	2.03	153.71	4.20	1.42	-116.48	1.64
7	6.62	109.83	22.61	1.96	171.32	4.14	1.42	-119.79	1.65
8	6.59	126.22	22.38	1.93	212.53	3.91	1.43	-125.49	1.67
9	6.54	145.82	22.19	1.86	234.78	3.82	1.43	-129.39	1.67
10	6.49	163.81	21.95	1.79	276.14	3.58	1.43	-132.93	1.67
11	6.47	182.94	21.74	1.76	301.24	3.47	1.42	-135.16	1.66
12	6.38	201.53	21.45	1.67	339.29	3.21	1.40	-136.63	1.64
13	6.35	222.71	21.16	1.60	372.58	3.09	1.38	-137.84	1.63
14	6.27	243.62	20.82	1.51	409.39	2.85	1.33	-129.86	1.55
15	6.18	265.35	20.44	1.47	445.40	2.70	1.26	-115.50	1.44
16	6.08	287.07	19.99	1.39	475.57	2.50	1.22	-104.31	1.38
17	6.00	311.29	19.52	1.26	521.74	2.20	1.17	-99.07	1.34
18	5.86	335.88	18.95	1.17	564.21	1.99	1.13	-80.96	1.27
19	5.73	359.63	18.34	1.08	605.87	1.82	1.08	-68.17	1.19
20	5.59	385.76	17.67	1.01	640.98	1.67	1.06	-52.48	1.15
21	5.45	410.44	16.96	0.95	681.85	1.50	1.01	-42.61	1.07
22	5.27	438.62	16.22	0.87	718.96	1.32	0.98	-30.70	1.04
23	5.06	466.03	15.43	0.80	757.25	1.17	0.96	-26.03	1.02
24	4.88	496.84	14.61	0.72	788.50	1.03	0.92	-25.38	0.97
25	4.67	525.27	13.70	0.66	827.84	0.91	0.91	-16.09	0.97

26	4.50	552.10	12.99	0.60	862.24	0.79	0.83	-14.48	0.91
27	4.39	568.08	12.58	0.54	901.06	0.70	0.80	5.39	0.89
28	4.18	596.19	11.80	0.49	932.66	0.60	0.71	14.25	0.81
29	3.94	630.70	10.85	0.44	966.35	0.54	0.65	49.67	0.76
30	3.74	661.86	10.05	0.40	1004.08	0.46	0.46	76.73	0.73
31	3.51	691.25	9.21	0.35	1024.00	0.41	0.34	131.56	0.71
32	3.31	718.83	8.47	0.31	1072.45	0.33	0.33	177.88	0.69
33	3.13	744.96	7.83	0.30	1086.04	0.34	0.30	193.65	0.69
34	2.95	767.44	7.25	0.27	1082.46	0.30	0.30	377.07	0.66
35	2.79	787.09	6.68	0.21	1116.14	0.20	0.32	521.96	0.71
36	2.67	808.08	6.27	0.17	1211.24	0.17	0.35	599.63	0.72
37	2.53	831.78	5.79	0.17	1239.07	0.15	0.37	669.38	0.66
38	2.43	844.10	5.46	0.18	1437.65	0.02	0.37	703.18	0.60
39	2.20	886.99	4.74	0.34	922.97	0.48	0.36	722.78	0.56
40	2.15	889.43	4.60	0.36	880.79	0.54	0.36	736.23	0.54