JOURNAL OF COMPUTATIONAL PHYSICS 24, 23–28 (1977)

A Method for Rapid Attainment of the Steady State in Two-Dimensional Time-Marching Supersonic Flow Calculations*

OTIS A. FARMER AND JOHN D. RAMSHAW

Group T-3, Theoretical Division, University of California, Los Alamos Scientific Laboratory, Los Alamos, New Mexico 87545

Received April 16, 1976; revised November 17, 1976

Finite-difference calculations of steady-state fluid flow problems are commonly performed by the asymptotic unsteady flow method, in which a time-dependent calculation is performed and the steady-state solution is approached asymptotically in the limit of long time. The purpose of this note is to present a simple method for reducing the amount of computer time required to perform such calculations in the case of primarily supersonic flow in two dimensions. In most cases, it should be a relatively simple matter to incorporate the method into existing two-dimensional time-marching compressible-flow computer codes. We are successfully using the method in conjunction with the RICE code [1] to calculate supersonic mixing flow in CW chemical lasers.

Consider the typical two-dimensional computing region shown in Fig. 1. The region is divided into cells of dimensions Δx and Δy and indexed by integers I and J in the x and y directions, respectively. There are IMAX cells in the x direction and JMAX cells in the y direction. The primary flow direction is the x direction, with the inflow at $I = 1$ $(x = 0)$ and the outflow at $I = \text{IMAX}$ $(x = L \equiv (\text{IMAX})\,\Delta x)$. The timestep is denoted by Δt and a representative flow speed is denoted by $\bar{u}$. The flow is assumed to be supersonic, so that physical boundary conditions are imposed at the inflow but not at the outflow.

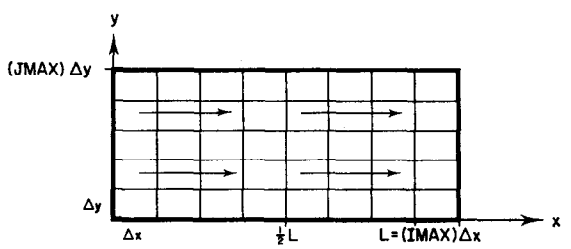

FIG. 1. Typical computing region and mesh.

* This work was supported by the Air Force Weapons Laboratory under project order 76-123, and was performed under the auspices of the United States Energy Research and Development Administration.

ISSN 0021-9991

We now consider the computer time required to perform a steady-state calculation in this region. Let the basic "grind time" (i.e., CPU time per cell per timestep) of the computer code in question be τ. The total CPU time required for the calculation is then

$$\text{CPU} = \tau(\text{IMAX})(\text{JMAX})\, N, \tag{1}$$

where N is the number of timesteps required. Let us assume, as is frequently the case, that N is proportional to the number of timesteps required for the fluid to traverse the mesh. Then

$$N = K(\text{IMAX})/\alpha, \tag{2}$$

where $\alpha = \bar{u}\, \Delta t/\Delta x$ is the convective Courant number and the coefficient of proportionality K is typically about 2.5. Combining Eqs. (1) and (2), we obtain

$$\text{CPU} = K\tau(\text{IMAX})^2(\text{JMAX})/\alpha. \tag{3}$$

Note that this expression is quadratic in IMAX.

The estimate of Eq. (3) applies to the usual straightforward procedure of calculating the entire computing region at once. But now consider the alternative procedure of dividing the region into two parts at the line $x = \frac{1}{2}L$ and computing each part to steady state separately. The upstream part ($0 < x < \frac{1}{2}L$) must of course be computed first. This calculation does not require outflow boundary conditions at $x = \frac{1}{2}L$ because the flow is supersonic. After the upstream section ($0 < x < \frac{1}{2}L$) has been calculated, the steady-state conditions at $x = \frac{1}{2}L$ are known and are to be used as the inflow boundary conditions for the downstream section ($\frac{1}{2}L < x < L$). The CPU time required to calculate either section is clearly given by Eq. (3) with IMAX replaced by IMAX/2. Thus the total CPU time required to calculate the two parts to steady state separately is

$$\text{CPU} = \tfrac{1}{2}K\tau(\text{IMAX})^2(\text{JMAX})/\alpha. \tag{4}$$

Comparison with Eq. (3) shows that by splitting the computing region into two parts we have reduced the CPU time required for the complete calculation by a factor of 2. This reduction is a consequence of the fact that Eq. (3) is quadratic rather than linear in IMAX. Similarly, it is clear that by splitting the region into n parts the CPU time required is reduced by a factor of n. Of course n cannot be chosen arbitrarily large, since in each part or subregion there must be enough cells in the x direction that numerical outflow boundary effects are negligible. The number of cells required for this purpose will depend, in general, upon both the problem under consideration and the numerical scheme, and must therefore be determined by numerical experimentation. In the RICE code, we have found in practice that the condition $n \lesssim (\text{IMAX})/10$ is sufficient; i.e., each subregion should contain about ten cells in the x direction. Thus for IMAX = 100 a savings in CPU time of a factor of 10 can be realized.

Although we have considered a rectangular region and mesh for purposes of

illustration, this is clearly not essential to the conclusions. It is also not essential that the flow be everywhere supersonic; it only needs to be supersonic along each of the lines at which the region is subdivided, so that in each subcalculation outflow boundary conditions are not required.

When first confronted by the above argument, one tends to feel that something has been obtained for nothing. However, a physical interpretation of the CPU time savings can readily be given. The usual straightforward approach of calculating the entire region at once is in fact highly inefficient, because the upstream cells become steady long before the downstream cells and continue to be calculated even though they are no longer changing. In addition, the downstream cells are calculated while cells upstream of them are still changing rapidly, before they have any possibility of becoming steady. By splitting up the region into several parts, the number of cells being calculated unnecessarily in this way is greatly reduced.

Although the procedure outlined above reduces the CPU time by a large factor, it is inconvenient to use because it requires one to perform a sequence of separate calculations, in which the steady-state outflow conditions from each calculation must be processed and converted into inflow boundary conditions for the next. A more convenient and systematic procedure is the following. In the computer code the region of calculation is specified by the range over which the I loops run, which in the basic code is from 1 to IMAX. It is a simple matter to modify the code so that these loops run from IL to IR, where IL and IR are allowed to change during the calculation to reflect the development of the steady region in the flow field. At the beginning of the calculation one would set IL = 1 and IR = IR1, where IR1 is a specified constant integer which would typically be about 10. (We emphasize again that this number must be determined empirically.) As soon as the column of cells with $I = 1$ becomes steady (according to some suitable criterion) one increments IL and IR by one, so that the I loops run from 2 to IR1 + 1. The fluid variables in the column $I = 1$ then remain frozen at their steady values and serve as the inflow boundary conditions for the region $I = 2$ to IR1 + 1. When the column $I = 2$ becomes steady IL and IR are again incremented by one, and so on. The calculation continues in this fashion until IR = IMAX, at which point IL and IR are held fixed and the calculation is continued until the column IMAX becomes steady. The CPU time required to perform the calculation in this manner is approximately the same as that required to split the calculation into IMAX/IR1 parts, namely

$$\text{CPU} = (\text{IR1}/\text{IMAX})\, K\tau(\text{IMAX})^2(\text{JMAX})/\alpha. \tag{5}$$

The presence of subsonic regions in the flow field is a complicating factor. It is necessary that the flow be supersonic in the entire column $I = \text{IR}$ at all times. If the subsonic regions are small, localized, and relatively few in number, this requirement can be met by monitoring the minimum Mach number in the column $I = \text{IR}$, and by incrementing IR by one whenever this number falls below unity. A subsonic region near $I = 1$ can be handled by choosing the initial IR large enough to enclose it, and then incrementing only IL until the difference IR–IL decreases to the desired value IR1 − 1, after which IL and IR are again incremented simultaneously.

At the beginning of the calculation initial conditions must be specified throughout the mesh. These initial conditions must be supersonic so that newly activated cells will be supersonic even when localized subsonic regions develop in the steady-state flow field.

In principle, the method described above is not applicable to fluids in which diffusion of mass, momentum, and energy may occur, because the equations then have a parabolic character and require outflow boundary conditions. In practice, however, this is not a restriction, because diffusion in the primary flow direction is in general negligible compared to convection in supersonic flow problems. (The ratio of diffusive to convective fluxes in the primary flow direction is of the order of the Knudsen number divided by the Mach number.) It therefore does no harm to simply use zero-gradient boundary conditions at the outflow at all times, so that the diffusive fluxes in the x direction are always zero at $I = \mathrm{IR}$. In this way the present method can be applied to flow problems in which diffusion of mass, momentum, and/or energy in the transverse direction is important. It is clear, however, that the method cannot be directly applied to viscous flow past a solid wall, because the flow becomes subsonic near the wall. For such problems the method could only be applied to the supersonic "outer" region of the flow, and would have to be coupled to a boundary-layer solution procedure for the flow near the wall. While this could be done it is not very convenient; thus the primary applicability of the present method is to flows of mixing-layer type, where no solid walls are present.

The above estimates of CPU time savings are based upon Eq. (2), which in effect identifies the characteristic time associated with the asymptotic approach to steady state with the time required for the fluid to traverse the mesh. In practice, of course, there are other characteristic times in the problem, and in the general case the longest of these times will govern the approach to steady state. In particular there exist transverse characteristic times associated with the extent of the mesh in the y direction. The most important of these is the time required for a sound signal to traverse the mesh in the y direction. In viscous or multicomponent flow problems there will also be a characteristic time associated with diffusion in the y direction. Ordinarily these transverse characteristic times are shorter than the longitudinal characteristic time upon which Eq. (2) is based, and hence do not govern the approach to steady state. However, when the mesh becomes very short in the x direction, as in the present method, these transverse characteristic times may eventually become longer than the longitudinal characteristic time. When this occurs Eq. (2) no longer applies and the CPU time savings predicted by Eq. (5) will be overly optimistic.

A simple test for steady conditions in the column IL which appears to work well is the following. Focus attention on a given cell in the column IL, and continuously monitor the value of the quantity q on the current timestep n and on the timestep $n - 5$. The given cell is considered to be steady when

$$\left| \frac{q^n}{q^{n-5}} - 1 \right| < 5a\alpha, \tag{6}$$

where $\alpha = \bar{u}\,\Delta t/\Delta x$ and the coefficient a is a convergence parameter. The column IL is considered to be steady when all cells in it are steady according to Eq. (6). This test cannot be applied until the column IL has run at least five cycles, which ensures that IL and IR cannot be incremented more frequently than every five cycles. Equation (6) is a difference approximation to the condition

$$|(1/q)(\partial q/\partial t)| < a(\bar{u}/\Delta x). \tag{7}$$

If the quantity q approaches its steady value exponentially with a time constant proportional to $\Delta x/\bar{u}$, then the value of a needed to ensure a given accuracy will be a constant, independent of the problem and flow conditions. If the time constant is determined by some other time scale in the problem, then a will be problem-dependent to a certain degree. A suitable value for a must be determined by numerical experimentation. We have obtained good results in supersonic mixing flow calculations by choosing pressure for the quantity q and setting $a \sim 10^{-3}$.

Finally, we illustrate the use of the method by considering a simple supersonic mixing flow calculation. The geometry is as shown in Fig. 1, with IMAX = 60, JMAX = 6, $\Delta x = 2.0$, and $\Delta y = 1.0$ (nondimensional units). Species B enters the region through the lower half of the inflow boundary ($y < 3.0$) with a velocity of 4.0, a density of 1.0, and a specific internal energy of 1.0. Species A enters the region through the upper half of the inflow boundary ($y > 3.0$) with a velocity of 2.0, a density of 0.5, and a specific internal energy of 2.0. Species A and B mix in the computing region and react chemically to produce a third species C, $A + B \rightarrow C$. This problem was run twice using the RICE code, first in the conventional manner (i.e., without marching) and second using the marching method described in this paper. The second (marching) calculation was performed using q = pressure, $a = 10^{-3}$, and IR1 = 10. The results of the two calculations agree to within one or two percent in species concentrations, as shown in Table I, and to better than one percent in the primary fluid dynamical variables (pressure, velocity, etc.). The first calculation required 353 sec of CDC 7600 CPU time, while the second calculation required only 44 sec.

TABLE I

Partial Density of Species C vs Transverse Distance y at $x = 119$ (I = 60) With and Without Marching Method

	Partial density of species C	
y	Without marching	With marching
0.5	0.014136	0.014372
1.5	0.064793	0.065538
2.5	0.22690	0.23040
3.5	0.52727	0.53145
4.5	0.40114	0.40363
5.5	0.25058	0.25566

ACKNOWLEDGMENT

We are grateful to T. D. Butler, M. C. Cline, C. W. Hirt, and W. C. Rivard for helpful discussions.

REFERENCES

1. W. C. RIVARD, O. A. FARMER, AND T. D. BUTLER, "RICE: A Computer Program for Multicomponent Chemically Reactive Flows at All Speeds," Los Alamos Scientific Laboratory report LA-5812, 1975.