JOURNAL OF COMPUTATIONAL PHYSICS 34, 348–370 (1980)

The Multigrid Method for Semi-Implicit Hydrodynamics Codes*

ACHI BRANDT

Department of Mathematics, Weizmann Institute of Science, Rehobot, Israel

AND

J. E. DENDY, JR., AND HANS RUPPEL

Theoretical Division, University of California, Los Alamos Scientific Laboratory, Los Alamos, New Mexico 87545

Received November 22, 1978; revised April 10, 1979

The multigrid method is applied to the pressure iteration in both Eulerian and Lagrangian codes, and computational examples of its efficiency are presented. In addition a general technique for speeding up the calculation of very low Mach number flows is presented. The latter feature is independent of the multigrid algorithm.

I. INTRODUCTION

In the simplest numerical schemes for hydrodynamics, all variables are advanced explicity in time. In such explicit schemes, the time step is restricted by the Courant condition, the limitation that sound signals propagate only a single cell per time step. In many situations such explicit schemes are prohibitively expensive. An example is the time evolution of the sun. A representative sound speed in the sun is 50 km/sec, varying, of course, with the depth into the sun. Such a sound speed implies signal transit times across the sun of several hours. However, a simulation of the sun's evolution as it proceeds toward the red giant stage would be concerned with time scales on the order of a billion years. Such a simulation would be impossible with a fully explicit scheme.

Hence, unless one is interested in resolving phenomena that occur on time scales of order $\Delta x/c$ (where $c\Delta t \leqslant \Delta x$ is the Courant condition), that is, unless accuracy conditions impose particularly stringent time steps one is led to consider schemes that treat at least some of the variable in an implicit-in-time fashion. The schemes considered in this paper treat the pressure implicitly. Often, of course, other aspects of the physical problem require implicit treatment: These may include diffusion

* This work was performed under the auspices of the USERDA. The U.S. Government's right to retain a nonexclusive royalty-free license in and to the copyright covering this paper, for governmental purposes, is acknowledged.

0021-9991/80/030348-23$02.00/0

Copyright © 1980 by Academic Press, Inc.
All rights of reproduction in any form reserved.

processes, chemical reactions, or even, in the above example of the sun, fusion reactions; these are not treated in this paper, though they are likely amenable to the same techniques developed here.

Implicit treatment of the pressure gives rise in general to a nonlinear elliptic equation, which may be solved with a variant of Newton's method coupled with overrelaxation. Given the impressive efficiency of the multigrid method [2], it is natural to consider its application in this framework. In Section 2 we discuss the application of the multigrid method to SOLA, a simple Eulerian, incompressible flow code; its simplicitly allows one to see more clearly features that might be obscured in more complicated codes. In Section 3 we consider the somewhat more complicated code SOLA-ICE, an Eulerian compressible flow code; in addition to the application of the multigrid method we discuss a technique that speeds up the calculation of very low Mach number flows with SOLA-ICE (or indeed with any code employing the ICE technique [9]). Finally, in Section 4 we study the problems associated with the application of the multigrid method to a Lagrangian code. Both Sections 3 and 4 have computational examples and timing comparisons of the multigrid method with simple overrelaxation. Those readers interested in the modified version of SOLA-ICE used in the numerical experiments may obtain a listing of the code upon request to the second author.

In order that this paper be reasonably self-contained, let us consider a brief description of the multigrid method applied to the equation

$$\Delta u = F \text{ in } \Omega = (0, 1) \times (0, 1), \qquad u = 0 \text{ on } \partial\Omega. \tag{1.1}$$

The motivation of the multigrid method is as follows. Traditional iterative techniques for Eq. (1.1) such as successive overrelaxation (SOR) do a fine job of reducing the high-frequency components of the error but a poor job of reducing the low-frequency components. By employing several grids one hopes to solve for the low-frequency components on a coarse grid, where the calculation is relatively inexpensive. In this context let us define a work unit as the work equivalent to a single iteration on the finest grid. This may be many iterations on the coarser levels. We solve for the high-frequency components on the finest grid when SOR is efficient. In this regard it turns out that $\omega = 1.0$ (Gauss–Seidel) is the most efficient choice of overrelaxation factor, since it is for this ω that one gets the best smoothing rate, that is, the most efficient reduction of the high-frequency components of the error. (For a definition of what constitutes high-frequency components, see [2].)

For simplicity we assume that we have only two grids, the coarse grid with mesh spacing h_c and the fine grid with mesh spacing $h_f = h_c/2$, the fine grid being obtained from the coarse grid by bisection. One begins with Gauss–Seidel on the fine grid for the equation

$$\Delta_{h_f} U^f = F, \tag{1.2}$$

where Δ_{h_f} is an approximation to the Laplacian on the fine grid. One iterates until convergence "slows down." Let u^f denote our approximate solution of (1.2), and let $V^f = U^f - u^f$ be our error. Slow convergence implies that V^f has become smooth, and

can therefore be approximated by a function V^c on the coarse grid. We thus switch to the coarse grid, solving there the equation

$$\Delta_{h_c} V^c = I_f^c(F - \Delta_{h_f} u^f), \tag{1.3}$$

where I_f^c is an interpolation from the fine grid to the coarse grid. Equation (1.3) is the coarse-grid approximation to the fine-grid equation for V^f,

$$\Delta_{h_f} V^f = F - \Delta_{h_f} u^f, \tag{1.4}$$

derived from Eq. (1.2). Let v^c be an approximate solution of Eq. (1.3) obtained, e.g., by Gauss–Seidel iterations on the coarse grid, starting with $v^c \equiv 0$. Then we correct our approximate solution u^f by

$$u_{\text{new}}^f \leftarrow u_{\text{old}}^f + I_c^f v^c, \tag{1.5}$$

where I_c^f is an interpolation operator from the coarse to the fine grid, so that $I_c^f v^c$ is an approximation to the error V^f. Thus, having liquidated high-frequency components of V^f on the fine grid we have efficiently reduced its low-frequency (smooth) components by the process outlined in Eqs. (1.3)–(1.5). This latter process is called the coarse-grid correction (CGC). We can then repeat this cycle, iterating on the fine grid and applying CGC alternately.

When we have more than two grids, the approximate solution of (1.3) may itself be obtained by combining a relaxation scheme on the coarse grid with CGC on a still coarser grid.

The above mode of the multigrid method is called the correction scheme (CS) [2]. In deriving (1.4) from (1.2) we used the linearity of our operator Δ_h. For a general nonlinear equation,

$$LU = F,$$

this cannot be done, and the suggested mode is the full approximation scheme (FAS) [2], which we now briefly describe. Again, one begins with Gauss–Seidel (or, better for the nonlinear case, Newton–Gauss–Seidel) iterations on the fine-grid equation

$$L_{h_f} U^f = F.$$

When convergence slows down, a coarse-grid correction is made, but now (1.3) is replaced by

$$L_{h_c} U^c = I_f^c(F - L_{h_f} u^f) + L_{h_c}(I_f^c u^f), \tag{1.6}$$

and (1.5) is replaced by

$$u_{\text{new}}^f \leftarrow u_{\text{old}}^f + I_c^f(u^c - I_f^c u^f), \tag{1.7}$$

where u^c is an approximate solution of (1.6). It is easy to see that if L is linear, CS and FAS are equivalent (u^c coinciding with $v^c + I_f^c u^f$). For further details the reader is referred to [2].

One final comment is that it is shown in [2] that when the multigrid method is applied to (1.2) the convergence factor, independent of the number of unknowns, is around 0.55. This is borne out by the examples displayed in Table I. (Note the constancy of the last column.) This is in contrast to the convergence rate for SOR with optimum ω applied to (1.2), which is approximately $1 - 2h_1$ [6]; thus, with SOR, the convergence factor deteriorates as the number of unknowns increases.

II. The Multigrid Method Applied to Incompressible Eulerian Hydrodynamics

The SOLA series of codes has won widespread acceptance as a tool for simulating a variety of two-dimensional, time-dependent fluid flows. Because the basic SOLA algorithm is constructed in a simple, straightforward fashion, it has been easy to adapt to other problem areas. For example, versions have been written to treat multiphase flow, coupled fluid-structure dynamics, free surfaces, curved boundaries, flow in porous media, and even three-dimensional problems. In this section and the next, we discuss the implementation of the multigrid method to the pressure iteration for two of the simpler members of this SOLA series: basic SOLA for incompressible flow and SOLA-ICE, which can treat all flow speeds.

In basic incompressible SOLA the equations solved are: the continuity equation

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{2.1}$$

and the Navier–Stokes equations

$$\begin{aligned} \frac{\partial u}{\partial t} + \frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} &= -\frac{\partial p}{\partial x} + g_x + \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right], \\ \frac{\partial v}{\partial t} + \frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} &= -\frac{\partial p}{\partial y} + g_y + \nu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right], \end{aligned} \tag{2.2}$$

where the velocity components, (u, v), are in the coordinate directions, (x, y); p is the ratio of pressure to density, the latter being constant in this incompressible limit. The external body forces are (g_x, g_y) and ν is the kinematic viscosity.

The finite-difference mesh used in SOLA is given in Fig. 1. The pressures are stored at cell centers and the velocities at cell sides, as shown in Fig. 2. If we lump the convection terms and the viscous accelerations, which are evaluated explicitly, into a single term on the right-hand side, then the difference equations in SOLA that approximate (2.1) and (2.2) are

$$D_{ij}^{n+1} \equiv (u_{i,j}^{n+1} - u_{i-1,j}^{n+1})/\Delta x + (v_{i,j}^{n+1} - v_{i,j-1}^{n+1})/\Delta y = 0, \tag{2.3}$$

$$\begin{aligned} u_{i,j}^{n+1} + \frac{\Delta t}{\Delta x}(p_{i+1,j}^{n+1} - p_{i,j}^{n+1}) &= a_{i,j}^{n}, \\ v_{i,j}^{n+1} + \frac{\Delta t}{\Delta y}(p_{i,j+1}^{n+1} - p_{i,j}^{n+1}) &= b_{i,j}^{n}. \end{aligned} \tag{2.4}$$

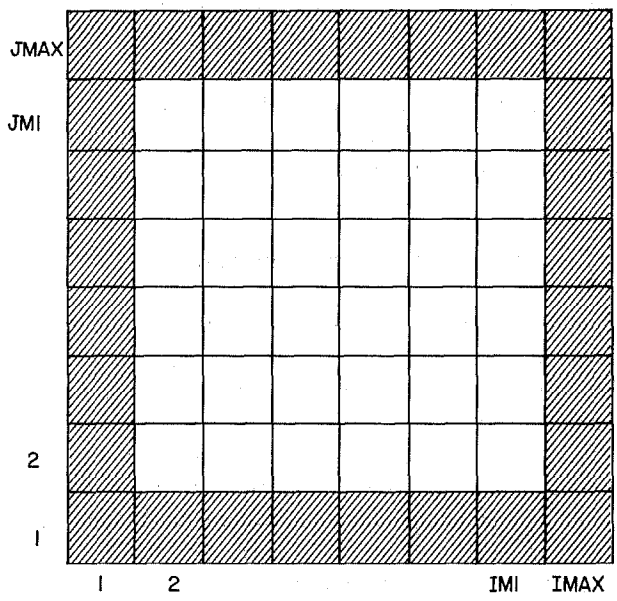

FIG. 1. Finite-difference mesh for SOLA.

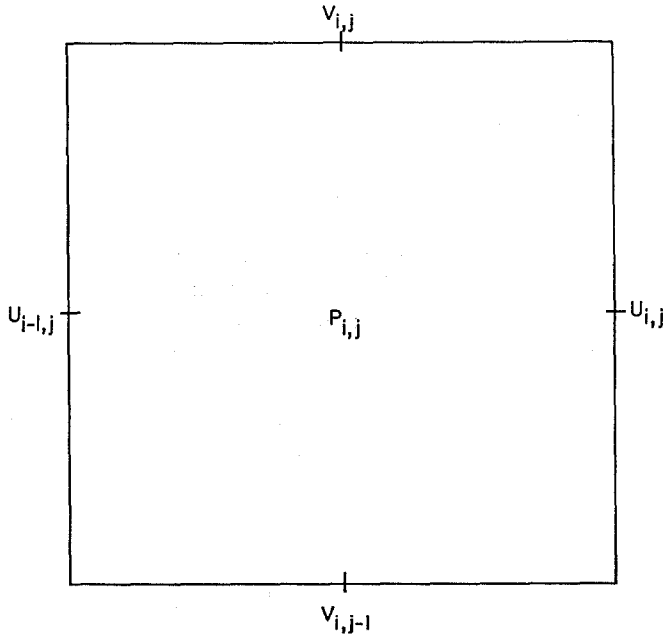

FIG. 2. Variable storage for SOLA.

The superscript, $n+1$, represents the advanced-time level of the dynamical variables After an explicit phase that guesses the advanced velocities, in terms of the old-time pressures and provides a starting point for the iteration, the above equations are solved by a variant of Newton's method.

Equation (2.3) is solved iteratively. For each cell, a pressure change $\delta p_{i,j}$ is obtained from the relation

$$\delta p_{i,j} = -\omega D_{i,j} \Big/ \left(\frac{\partial D_{i,j}}{\partial p_{i,j}}\right). \tag{2.5}$$

By applying a virtual pressure change, one can derive from (2.3) and (2.4) that $\partial D_{ij}/\partial p = 2\Delta t(1/\Delta x^2 + 1/\Delta y^2)$. The quantity ω is merely an overrelaxation factor.

Having obtained δp from (2.5) one uses it to update the velocities; thus, the iterative procedure is the following [8] (the tildes represent partially advanced quantities as the sweep proceeds through the mesh).

$$\begin{aligned}
\tilde{u}_{i,j}^{(k)} &= u_{i,j}^{(k)} + \frac{\Delta t}{\Delta x}\,\delta p_{i,j}^{(k)}, \\
u_{i-1,j}^{(k+1)} &= \tilde{u}_{i-1,j}^{(k)} - \frac{\Delta t}{\Delta x}\,\delta p_{i,j}^{(k)}, \\
\tilde{v}_{i,j}^{(k)} &= v_{i,j}^{(k)} - \frac{\Delta t}{\Delta y}\,\delta p_{i,j}^{(k)}, \\
v_{i,j-1}^{(k+1)} &= \tilde{v}_{i,j-1}^{(k)} - \frac{\Delta t}{\Delta y}\,\delta p_{i,j}^{(k)}, \\
\delta p_{i,j}^{(k)} &= \omega\left(\frac{1}{\Delta x}(\tilde{u}_{i-1,j}^{(k)} - u_{i,j}^{(k)}) + \frac{1}{\Delta y}(\tilde{v}_{i,j-1}^{(k)} - v_{i,j}^{(k)}) + d_{i,j}\right)\Big/\left(2\Delta t\left(\frac{1}{\Delta x^2} + \frac{1}{\Delta y^2}\right)\right), \\
p_{i,j}^{(k+1)} &= p_{i,j}^{(k)} + \delta p_{i,j}^{(k)},
\end{aligned} \tag{2.6}$$

where $k \geqslant 0$, $2 \leqslant i \leqslant IM1$, $2 \leqslant j \leqslant JM1$. Here $d_{i,j} = 0$ but will be nonzero on the coarser grids in the multigrid method.

Although (2.6) was derived from Newton's method, it is in fact a restructured version of the traditional SOR method for the equation

$$\begin{aligned}
Lp_{i,j} &= -p_{i-1,j} - p_{i+1,j} - p_{i,j-1} - p_{i,j+1} + 4p_{i,j} \\
&= F_{i,j} \equiv \frac{1}{\Delta t}(\Delta x(a_{i,j}^n - a_{i-1,j}^n) + \Delta y(b_{i,j}^n - b_{i,j-1}^n)),
\end{aligned} \tag{2.7}$$

where a^n and b^n are given in Eq. (2.4). This fact is implicitly stated in [11], and its demonstration is given in the Appendix. A measure of how well (2.6) is solved is the quantity

$$G_{ij}^{(k+1)} = F_{ij} - 4p_{i,j}^{(k+1)} + p_{i-1,j}^{(k+1)} + p_{i,j-1}^{(k+1)} + p_{i+1,j}^{(k)} + p_{i,j+1}^{(k)}, \tag{2.8}$$

which, as is also shown in the Appendix, is a multiple of $D^{(k)}$; reduction of $D^{(k)}$ below some fixed tolerance is the convergence criterion employed in SOLA.

The decision to use (2.6) instead of SOR for (2.7) is arbitrary in the incompressible case; however, in the compressible case of the next section, (2.6) readily allows for a generalization of the equation of state for the pressure, whether given analytically or by tables.

Let us consider now the implementation of the multigrid method to (2.6). We need a nested sequence of staggered grids, the first two levels of which are depicted in Fig. 3 and which, for simplicity, we refer to as the fine and coarse grids.

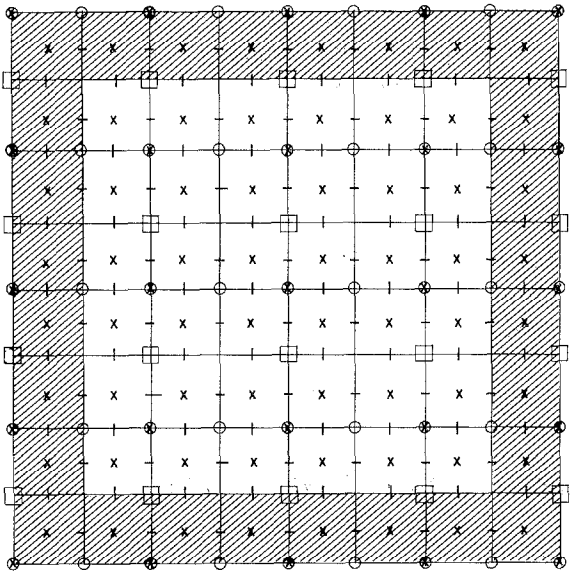

FIG. 3. Two grids for multigrid for SOLA.

The fine-grid p's are denoted by crosses, and the fine-grid u's and v's are denoted by horizontal and vertical hash marks, respectively. The coarse-grid p's are denoted by circled crosses. The coarse-grid u's and v's lie on intersections of grid lines; and intersections where the u's [v's] lie are enclosed with circles [squares]. Note that the coarse-grid p's, u's, and v's are located at points different from the location of the fine grid p's, u's, and v's, so that coarse-grid quantities have to be obtained by averaging fine-grid quantities. This is not a drawback; in fact, as we explain below, averaging is desirable.

In our discussion of the implementation of the multigrid method, we will consider the algorithm for two grids only; this simplifies the description and includes the main features of the method. The algorithm we describe corresponds to the "correction scheme" of multigrid [2], since this is a linear problem.

One begins with (2.6) on the fine grid with $\omega = 1$, to provide the best smoothing rate, and iterates until convergence slows down. A switch is then made to the coarser grid by computing

$$D_{i,j}^{\text{fine}} \equiv D_{i,j}^{\text{f}} = \left(\frac{1}{\Delta x}(u_{i,j}^{\text{f}} - u_{i-1,j}^{\text{f}}) + \frac{1}{\Delta y}(v_{i,j}^{\text{f}} - v_{i,j-1}^{\text{f}})\right)$$

and forming $D_{kl}^{\text{c}} \equiv D_{kl}^{\text{coarse}}$ by averaging the four nearest $D_{i,j}^{\text{f}}$'s. Each time the iteration drops to the coarser grid, u, v, and p are set to zero and the iteration of Eq. (2.6) proceeds, except that now $d_{i,j} = -D_{i,j}^{\text{c}}$ in (2.6).

The way to understand the origin of the term $d_{i,j}$ is to recall that the equation to be solved on the fine grid is $D_{i,j}^{\text{f}} = 0$. The residual of this equation at a given stage is $-D^{\text{f}}$, and d is simply $-I_{\text{f}}^{\text{c}} D^{\text{f}}$; the u^{c}, v^{c}, and p^{c} obtained from solving $D^{\text{c}} = d$ act as corrections to the current u^{f}, v^{f}, and p^{f} and serve to reduce the residual on the fine grid. This is analogous to the procedure described in the Introduction for $\Delta u = F$.

Consider several levels of coarse grids: If the iteration is proceeding at a given level of coarseness, two possibilities exist. If a predetermined convergence criterion is met, the u's, v's, and p's are interpolated to the next-finer grid and added to their most recent values on this finer grid. The iteration then shifts to this finer grid. On the other hand, if the iteration on the given level of coarseness slows down, the iteration drops to the next-coarser level, as outlined above. At the coarsest possible level, the convergence criterion must be met, or the calculation terminates after some preassigned amount of work.

As mentioned in the discussion of Fig. 3, the construction of the coarse grid from the fine grid for the SOLA-like arrangement of variables requires that the dynamical variables be located at mesh positions other than those occupied on the fine grid. We will refer to this situation as one in which the coarse-grid points are not a subset of the fine-grid points; that is, they must be obtained by interpolation. In [2], for Laplace's equation with Dirichlet boundary conditions and for the case in which coarse-grid points are a subset of fine-grid points, the right-hand side of Eq. (2.6) on the coarse mesh at a given point was given by the residual of this equation on the fine mesh at that point. Brandt discovered later [3] that for Neumann boundary conditions, this prescription leads to a degradation of around 40 % in the convergence rate. A remedy he proposed is to use a weighted average of neighboring residuals. Because in our circumstance the coarse-grid points are not a subset of fine-grid points, some such averaging is forced upon us. We simply weight the four neighboring residuals of Eq. (2.6) equally with weight 0.25. However, near the boundary an unequal weighting must be used; otherwise there is a degradation in convergence rate of around 10 %. The reason for this is that the pressures near the left boundary, for example, are $\Delta x/2$ away from the boundary and not Δx.

Another new feature of our treatment concerns the manner of interpolation as one moves from the coarse grid to the fine. For definiteness consider the problem of obtaining the velocity, u, in the SOLA configuration, if one imposes a no-slip condition at the top boundary. Naive considerations might lead one to extrapolate linearly from the neighboring u's, to obtain a u-velocity adjacent to the boundary that leads to the

desired no-slip condition. Separate linear interpolation would also be done for the v-velocity and the pressure p.

This straightforward procedure leads to incorrect results for the pressure iteration since it in general destroys the relationships (2.4). The effect of this procedure is to introduce a perturbation to the right-hand side of (2.7) such that

$$Lp = F + \epsilon$$

is the equation that is being solved instead of (2.7); the effect is the introduction of errors that never get smoothed out. A correct interpolation procedure is to interpolate the p^c's bilinearly to obtain δp^f and then to perform the replacements

$$p^f_{i,j} \leftarrow p^f_{i,j} + \delta p^f_{i,j},$$

$$u^f_{i,j} \leftarrow u^f_{i,j} + \frac{\Delta t}{\Delta x}(\delta p^f_{i,j} - \delta p^f_{i+1,j}),$$

$$v^f_{i,j} \leftarrow v^f_{i,j} + \frac{\Delta t}{\Delta y}(\delta p^f_{i,j} - \delta p^f_{i,j+1}),$$

so that relationships (2.4) are preserved. For the SOLA algorithm the two procedures are the same everywhere but at the boundary; however, in some cases (see Section 4), they are different everywhere. Careful studies have been performed with SOLA on the test problem given in the report, viscous driven flow in a cavity. When the boundary problem is correctly handled, the convergence factor for the pressure iteration is a little less than the value of 0.6 predicted [2].

III. The Multigrid Method Applied to Compressible Flow in an Eulerian Mesh

We will now consider the application of the multigrid method to compressible flow by implementing it in the Eulerian code, SOLA-ICE. Although the equations involved are similar to those solved in SOLA, the nonconstant density and the addition of the effects of temperature on the pressure through an equation of state introduce certain differences into the iteration. For completeness we include the full set of equations as solved in SOLA-ICE.

For mass conservation we write

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

The momentum equations are

$$\rho\left(\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = \rho g_x - \frac{\partial p}{\partial x} + \mu\left(\frac{4}{3}\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{1}{3}\frac{\partial^2 v}{\partial x\,\partial y}\right)$$

and

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=\rho g_y-\frac{\partial p}{\partial y}+\mu\left(\frac{4}{3}\frac{\partial^2 v}{\partial y^2}+\frac{\partial^2 v}{\partial x^2}+\frac{1}{3}\frac{\partial^2 u}{\partial x\,\partial y}\right).$$

Although they do not concern us directly, note that the compressibility feature introduces differences also in the form of the viscous terms. The internal energy equation is

$$\begin{aligned}\rho\left(\frac{\partial I}{\partial t}+u\frac{\partial I}{\partial x}+v\frac{\partial I}{\partial y}\right)&\\ =-p\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)+K\left(\frac{\partial^2 T}{\partial x^2}+\frac{\partial^2 T}{\partial y^2}\right)&\\ +\mu\left[2\left(\frac{\partial u}{\partial x}\right)^2+2\left(\frac{\partial v}{\partial y}\right)^2+\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right)^2-\frac{2}{3}\left(\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\right)^2\right],&\end{aligned}$$

with $I = C_v T$. The equation of state is

$$p = F(\rho, I) = a^2(\rho - \rho_0) + (\gamma - 1)\rho I.$$

As in SOLA one first uses a completely explicit procedure to estimate the advanced-time velocities via a discrete version of the momentum equations. The iteration procedure described below is then employed to solve the equation of state to some tolerance. This yields approximations to the advanced-time pressure and velocity fields which are used to advance the density and internal energy via discrete versions of the density and internal energy equations. The specific difference equations employed are given in [4]. We write

$$W \equiv \bar{p} - F(\tilde{\rho}, \bar{I}) = 0;$$

the density $\tilde{p}$ depends on the most advanced volumes, which in turn depend on the velocities. The internal energy is updated only by the pdV work. The viscous terms are not included in this estimate of $\bar{I}$; they are treated purely explicitly. To solve this equation we use a variant of Newton's method approximating the Jacobian, $\partial W/\partial p$, numerically. Experience has shown that for most problems the Jacobian need be updated only once per cycle, at the end of the explicit guess.

The storage of the dynamical variables is the same as in SOLA, with the addition that ρ and I are stored at cell centers along with the pressures. The iteration, then, proceeds as follows [4]. For each cell one calculates the divergence of the velocity for that cell with the most current information available, that is,

$$D_{i,j}^{(k)} = \frac{1}{\Delta x}(\tilde{u}_{i,j}^{(k)} - u_{i-1,j}^{(k)}) + \frac{1}{\Delta y}(\tilde{v}_{i,j}^{(k)} - v_{i,j-1}^{(k)}).$$

From the Lagrangian form of the continuity equation one can solve for the density to obtain

$$\bar{\rho}_{i,j}^{(k)} = \rho_{i,j}^{(k)}/(1 + \Delta t\, D_{i,j}^{(k)}).$$

This represents the effect of volume changes on the density. The convection is treated separately in terms of the advanced velocities. Calculating the change in internal energy from work done on the cell (i.e., due to volume changes) gives

$$\bar{I}_{i,j}^{(k)} = I_{i,j}^{(k)} - \Delta t\, \bar{p}_{i,j}^{(k)} D_{i,j}^{(k)}/\rho_{i,j}^{n}.$$

Then from Newton's method the pressure change can be obtained, yielding

$$\delta p_{i,j}^{(k)} = \frac{-\omega}{(\partial W/\partial \bar{p})_{i,j}} [\bar{p}_{i,j}^{(k)} - F(\bar{\rho}_{i,j}^{(k)}, \bar{I}_{i,j}^{(k)})]$$

and finally the pressure

$$\bar{p}_{i,j}^{(k+1)} = p_{i,j}^{(k)} + \delta p_{i,j}^{(k)}.$$

With the pressure change, the four cell velocities can be updated.

$$\begin{aligned}
\tilde{u}_{i,j}^{(k)} &= u_{i,j}^{(k)} + \frac{2\Delta t}{\Delta x}\, \delta p_{i,j}^{(k)}/(\rho_{i,j}^{n} + \rho_{i+1,j}^{n}),\\
u_{i-1,j}^{(k+1)} &= \tilde{u}_{i-1,j}^{(k)} - \frac{2\Delta t}{\Delta x}\, \delta p_{i,j}^{(k)}/(\rho_{i,j}^{n} + \rho_{i-1,j}^{n}),\\
\tilde{v}_{i,j}^{(k)} &= v_{i,j}^{(k)} + \frac{2\Delta t}{\Delta y}\, \delta p_{i,j}^{(k)}/(\rho_{i,j}^{n} + \rho_{i,j+1}^{n}),\\
v_{i,j-1}^{(k+1)} &= \tilde{v}_{i,j-1}^{(k)} - \frac{2\Delta t}{\Delta y}\, \delta \rho_{i,j}^{(k)}/(\rho_{i,j}^{n} + \rho_{i,j-1}^{n}).
\end{aligned} \tag{3.1}$$

One can combine the above equations to obtain a single equation for the time-advanced pressure in terms, solely, of old-time quantities. If we define $h \equiv \Delta x \equiv \Delta y$, we may write at convergence that

$$D_{i,j}^{(k)} \simeq D_{i,j}^{(k-1)} - \frac{\Delta t}{\rho_{i,j}^{n}} \Delta_h\, \delta p_{i,j}^{(k-1)} \simeq D_{i,j}^{n} - \frac{\Delta t}{\rho_{i,j}^{n}} \Delta_h p_{i,j}^{(k)}, \tag{3.2}$$

$$p^{n+1} \equiv p^{(k+1)} \simeq p^{(k)},$$

$$\begin{aligned}
\bar{p}^{(k)} &\simeq F(\bar{\rho}^{(k)}, \bar{I}^{(k)})\\
&= a^2 \left(\frac{\rho^n}{1 + \Delta t\, D^{(k)}} - \rho_0\right) + (\gamma - 1) \frac{\rho^n}{(1 + \Delta t\, D^{(k)})} \left(I^n - \Delta t \frac{\bar{p}^{(k)} D^{(k)}}{\rho^n}\right),
\end{aligned} \tag{3.3}$$

from which it follows that

$$(1 + \Delta t\, D^{(k)})\, \bar{p}^{(k)} = a^2\rho^n - a^2\rho_0(1 + \Delta t\, D^{(k)}) + (\gamma - 1)\, \rho^n I^n - \Delta t(\gamma - 1)\, \bar{p}^{(k)} D^{(k)}.$$

Using (3.2) and (3.3) yields

$$-\left(\frac{\Delta t^2}{\rho_n}\gamma p^{n+1} + \frac{a^2\rho_0}{\rho^n}\Delta t^2\right)\Delta p^{n+1} + (1 + \gamma\, \Delta t\, D^n)\, p^{n+1} = a^2\rho^n - a^2\rho_0(1 + \Delta t\, D^n) + (\gamma - 1)\, \rho^n I^n. \tag{3.4}$$

Since this is an elliptic equation, albeit nonlinear, there should be no problem with the application of the multigrid method; however, the nonlinearity forces one to use the FAS mode (full approximation storage mode) instead of the correction mode [2]. One uses the basic algorithm (3.1) on the finest grid; when convergence slows down one switches to the coarse grid. (Again we assume only two grids for ease of exposition.) One lays down on the coarse mesh the fine-mesh values of u, v, p, ρ, I. In our case this requires averaging since once again the coarse mesh is not a subset of the fine mesh. Having done the averaging onto the coarse mesh one forms the residuals of W,

$$r^{\mathrm{f}}_{i,j} = f(\bar{\rho}^{\mathrm{f}}_{i,j}, \bar{I}^{\mathrm{f}}_{i,j}) - \bar{p}^{\mathrm{f}}_{i,j} \qquad \text{and} \qquad r^{\mathrm{c}}_{k,l} = f(\bar{\rho}^{\mathrm{c}}_{k,l}, \bar{I}^{\mathrm{c}}_{k,l}) - \bar{p}^{\mathrm{c}}_{k,l}.$$

In the interior of the coarse mesh,

$$R^{\mathrm{c}}_{k,l} = -r^{\mathrm{c}}_{k,l} + \tfrac{1}{4}(r^{\mathrm{f}}_{i,j} + r^{\mathrm{f}}_{i+1,j} + r^{\mathrm{f}}_{i,j+1} + r^{\mathrm{f}}_{i+1,j+1}),$$

where k, l, i, j are such that the coarse-grid point labeled by k, l is in the center of the rectangle formed by the fine-grid points labeled (i, j), $(i + 1, j)$, $(i, j + 1)$, $(i + 1, j + 1)$. Near the boundaries a different average of fine residuals is used. On the coarse mesh one uses the algorithm (3.1) except that now $W^{\mathrm{c}} \equiv \bar{p} - F(\bar{\rho}, \bar{I}) - R^{\mathrm{c}} = 0$ is the equation that one wishes to satisfy and accordingly the pressure change is

$$(\delta p^{\mathrm{c}}_{i,j})^{(k)} = \frac{-\omega}{(\partial W^{\mathrm{c}}/\partial \bar{p})_{i,j}}[(\bar{p}^{\mathrm{c}}_{i,j})^{(k)} - F(\bar{\rho}^{\mathrm{c}}_{i,j}, \bar{I}^{\mathrm{c}}_{i,j}) - R^{\mathrm{c}}_{i,j}]. \tag{3.5}$$

When the multigrid method was first tried in SOLA-ICE, it performed well except for spending an inordinately long time on the coarsest grid. The reason for this phenomenon is not hard to discover. We take as a simplified model of (3.4) the equation

$$-\Delta p + \sigma p = f \text{ in } \Omega, \qquad \frac{\partial p}{\partial \nu} = 0 \text{ on } \partial\Omega, \tag{3.6}$$

where σ is a positive constant. A local mode analysis [2] shows that the convergence factor of Gauss–Seidel applied to (3.6) can be no better than $s = 2/(2 + \sigma h^2)$, the

factor by which the constant part of the error is reduced each sweep. As $\sigma \to 0$, $s \to 1$. Hence for small σ, Gauss–Seidel has to work very hard to determine the constant part of the solution. It is interesting that this problem does not arise when $\sigma = 0$, since in that case the solution is determined only up to a constant anyway, and the convergence rate, which is determined by the frequency nearest $\theta_1 = 0$, $\theta_2 = 0$, is quite good, at least for coarse meshes. Nor does the problem arise when at least one boundary has a Dirichlet condition imposed, since there is more than just σ to tie down the solution in that case. In other words, a basic premise of the multigrid method—that the problem is easy to solve on the coarset grid—is violated for (3.6) if σ is small.

An easy way to avoid the above problem for (3.6) is to employ a direct solution on the coarsest grid. This remedy is not easy for Eq. (3.4), however, and in any case it puts a limitation on the size of the coarsest grid because of storage considerations, especially if one contemplates doing three-dimensional problems. A remedy due to Brandt in terms of Eq. (3.6) is to form

$$C = \sum_{i,j}{}' \left(f_{ij}^{\text{coarse}} + \Delta_{h_c}(p_{i,j}^c)^{(k)} - \sigma(p_{i,j}^c)^{(k)}\right) \Big/ \sum_{i,j}{}' \sigma$$

after every Gauss–Seidel iteration on the coarse grid and to add this number to $(p^c)^{(k)}$ at every grid point. (The prime in the summation denotes a weighted sum. The weights are different from 0.25 only near the boundary; using a weighted sum helps most when σ has jump discontinuities.) This method is derived by substituting $(p^c)^{(k)} + C$ for $(p^c)^{(k)}$ and solving for C. In the case of (3.5) this leads to the prescription

$$\left(\sum_{i,j}{}' 1\right) C = \sum_{i,j} (f(\bar{\rho}_{i,j}^c, \bar{I}_{i,j}^c) - \bar{p}_{i,j}^c - R_{i,j}^c)/(1 + \Delta t\, D_{i,j}^c). \tag{3.7}$$

Fortunately the $1 + \Delta t\, D$ contribution can be ignored in (3.7) with little loss in efficiency. That is, one can use instead

$$\left(\sum_{i,j}{}' 1\right) C = \sum_{i,j}{}' (f(\bar{\rho}_{i,j}^c, \bar{I}_{i,j}^c) - \bar{p}_{i,j}^c - R_{i,j}^c).$$

This is fortunate because equations of state are frequently not known explicitly but are given in terms of tables, in which case derivations like the one leading to (3.4) cannot be performed. An equation of state given by tables would also preclude deriving an equation like (3.4) in general if one were inclined to compute with the elliptic equation one was actually solving instead of using (3.1).

The Mach number of the flow is proportional to σh^2 assuming $u\delta t/h \approx 1$; hence, this effect is especially pronounced, and the constant determination especially important, for low Mach number flows.

We ran several comparisons with the single-grid and multigrid versions of SOLA-ICE. The one we present here is that of a thermally driven convection in a cylindrical can heated on the vertical boundary. This problem is described in more detail in [4, p. 15]. Since the Mach number in this case is quite small, 10^{-4}, this problem presents

a real challenge to SOLA-ICE. Indeed, the mesh used in [4] for this problem was only 8×8; for a 12×12 mesh the version in [4] takes 1000 iterations to reduce an initial error of 1.856×10^{4} to 6.263×10^{-1}; since the convergence criterion asked for is 5.0×10^{-8}, it is clear that the version in [4] cannot handle this problem on a mesh much larger than 8×8. The problem, of course, is that the equation being solved is (3.4), with $a^2 = 0.0$, $\gamma = 1.4$, $\Delta t = 5.0 \times 10^{-1}$, $\rho_0 = 1.0 \times 10^{-4}$, and $p^0 \simeq 8.0 \times 10^{4}$, so that $(\Delta t^2/\rho^0)\,\gamma p^0 \simeq 56.0$; this is like having $\sigma = 1/56$ in (3.6). The addition of the constant described above enables one to perform the calculation on a 12×12 grid, with about 100 iterations to reduce the error below 5.0×10^{-8}. In our comparison we allow the single-grid algorithm to use the addition of the constant since the addition of the constant is not inherently connected with the multigrid method and since the comparison of the two methods is ludicrous otherwise. In this regard we note that incorporating the addition of the constant routine into a code is a very easy thing to do, much easier than incorporating the whole multigrid method.

With the single-grid method, there is an overrelaxation parameter ω to choose; ω is usually chosen in an ad hoc manner and does not bear much resemblance to the optimum ω for the model problem, $\Delta u = f$ [10]. In this case we found the recommended ω of 1.5 of [4] to be nearly optimal for both the 12×12 and the 24×24 grids. The comparisons shown in Table I consisted of running the problem for 10 time steps and comparing the time for iteration and overall time for the multigrid versus the single-grid method. The time step for the 24×24 problem was taken to be 2.5×10^{-4}, half that of the 12×12 problem; the time step was such that the multigrid method exhibited a convergence rate of a little less than 0.6. (With a sufficiently small time step one would expect only high-frequency errors and a convergence rate as good as the smoothing rate, 0.5. In this case the multigrid method would not use the coarse grids at all. For a sufficiently large time step, one would expect the quasi-Newton iteration's convergence rate to dominate the process, and in this case one would not obtain a convergence rate as good as 0.6.)

TABLE I

Size of problem	Number of grids used (Multigrid)	Total time spent iterating (MG/SG)	Total calculational time (MG/SG)	Fraction of calculation spent Iterating (SG)	Fraction of calculation spent Iterating (MG)
12×12	3	0.35	0.61	0.60	0.34
24×24	4	0.12	0.29	0.80	0.33

Note that the basic iteration is based on point Gauss–Seidel. If Δx were more than 2 times Δy, or vice versa, one would have to modify the algorithm to incorporate line Gauss–Seidel in order to recover a good smoothing rate and therefore a good convergence rate for the multigrid method [2].

A final comment is that the multigrid algorithm employed here is not the only possible one. An attractive alternative is the one described in [2, Sect. 6.3]; this algorithm gives a procedure for solving a problem (Poisson's equation with the five-point operator [2]) to within truncation error in a fixed number (around five) of work units. This is more efficient than solving to within a specified tolerance, since the tolerance is usually chosen either greater or less than truncation error; if greater, one is not getting the accuracy that is possible; if less, one is presumably wasting work. Of course, the success of the algorithm in the context of the pressure iteration could be determined only empirically. Note, however, than any improvement via the new algorithm would be marginal. The present algorithm reduces the fraction of time spent in the pressure iteration from a fraction which approaches 1 as the number of unknowns increases to a fraction that remains constant. For the test problem this constant was $\frac{1}{3}$; hence, even if one reduced the pressure iteration time to zero, this would give only 30 % savings in time, and since the test problem is harder than most typical problems, the savings would usually not even be as great as 30 %.

IV. The Multigrid Method Applied to Compressible Lagrangian Hydrodynamics

Consideration of the application of the multigrid method to Lagrangian hydrodynamics actually precedes the work of the previous sections [5]. This early work began with the straightforward application of the multigrid method in a finite-element environment; that is, the fine grid was obtained by beginning with a coarse gird and successively bisecting it. Following success in this situation the method was examined in the context of a Lagrangian grid, in which one is given the fine mesh and has to make sense out of a coarser mesh. A naive way of doing this is just to pluck out every other line of the fine grid to obtain the coarse grid.

Thus in Fig. 4, the fine grid is given by solid lines while the coarse grid is given by dotted lines. The experiments with the finite-element method used Laplace's equation and piecewise continuous bilinear elements defined on the elements formed by the vertices in Fig. 4; the quadratures were performed using four Gauss points. Initially it was thought that the interpolation required relatively sophisticated techniques such as bilinear interpolation; however, it was discovered that the simplest interpolation (in which, for example, E is declared by fiat to be the centroid of quadrilateral $ABCD$) worked just as well if not better. A heuristic explanation of this phenomenon is not hard to find. One can view $-\Delta u$ on the distorted mesh as being $-\nabla \cdot (D\nabla \mathbf{u})$ on a rectangular mesh with a widly varying D. Since numerical experiments by Brandt indicate that the multigrid method performs well even when D is random (but positive), the simple interpolation is at least heuristically justified.

The success of these initial experiments led to the search for an application in a Lagrangian hydrodynamics code. The code which was chosen was SALE, a simplified version of YAQUI [1]. This code has many features in common with SOLA-ICE. As in SOLA-ICE one advances the velocities explicitly and then solves the equation of

state implicitly. Now, of course, the cell vertices move with the fluid, and this gives rise to distorted meshes. In contrast to SOLA-ICE, the velocities u and v are stored at cell vertices. Let us remark that the interpolation from coarse to fine grid is performed analogously to the interpolation in SOLA; that is, there are two relationships analogous to (2.4) which must be preserved, and in this case bilinear interpolation of u, v, and p is globally different from interpolating the pressures and using them to preserve the relationships analogous to (2.4).

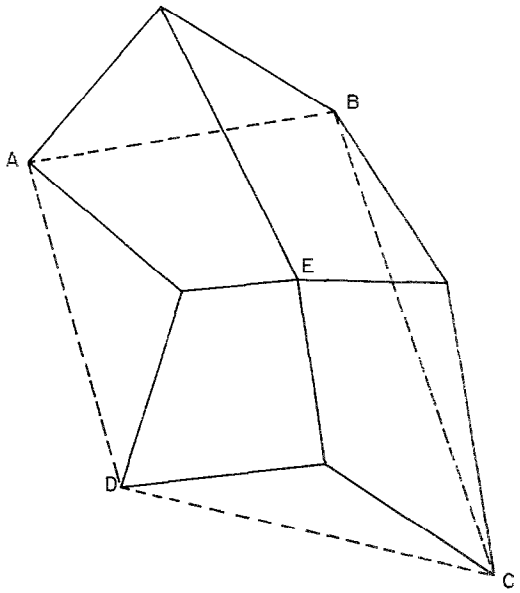

FIG. 4. Two grids for multigrid for SALE, fine grid given by solid lines, coarse grid by dotted lines.

The equation of state is as in SOLA-ICE and the elliptic equation that is being solved at each time step is (3.4). If we view the equation we are solving as an equation on the logical mesh, the coefficients can be wildly varying (if the physical mesh is distorted), and the multigrid method requires the coefficients on the coarse grids to be obtained by averaging the coefficients on the finest grid. But this averaging process is built into our situation since we are forced to average the fine-grid pressures to obtain the coarse-grid pressures.

With so many similarities between SALE and SOLA-ICE, it came as somewhat of a surprise to discover that the multigrid method worked no better than SOR in SALE, especially in the light of the early successful numerical experiments using the finite-element method on Lagrangian meshes. The reason for this failure is apparent if one examines the differencing in SALE [1].

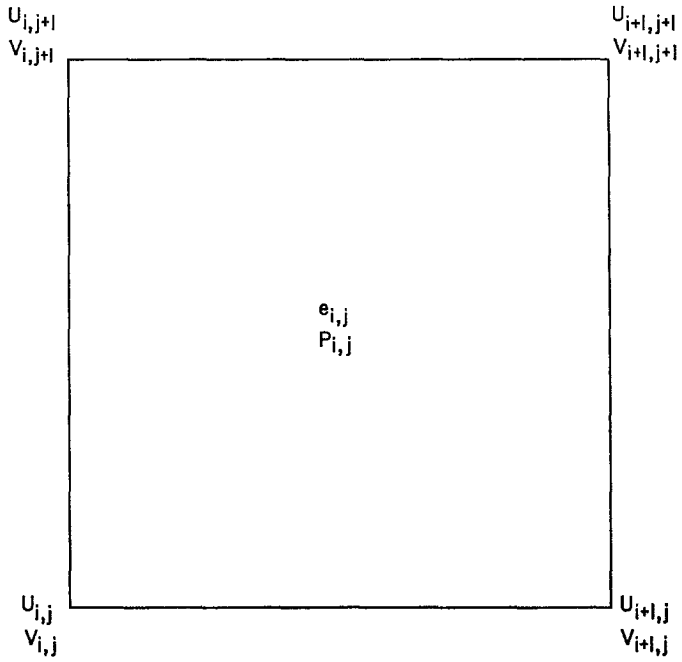

FIG. 5. Variable storage for SALE.

To explain this we simplify and consider only a rectangular grid; the variable storage is shown in Fig. 5. Since u's and v's are stored at cell vertices the divergence D_{ij} reduces to

$$\frac{1}{2\Delta x}(u_{i+1,j+1} + u_{i+1,j} - u_{i,j+1} - u_{ij}) + \frac{1}{2\Delta y}(v_{i+1,j+1} + v_{i,j+1} - v_{i+1,j} - v_{i,j})$$

in this special case; that is, averaging is necessary. The updating of u, for example, also requires averaging:

$$u_{i,j}^{(k+1)} = u_{i,j}^{(k)} + \frac{\Delta t}{2\Delta x}(\delta p_{i,j}^{(k+1)} - \delta p_{i-1,j}^{(k)} + \delta p_{i,j-1}^{(k+1)} - \delta p_{i-1,j-1}^{(k)})/$$

$$(\tfrac{1}{4}(\rho_{i,j} + \rho_{i-1,j} + \rho_{i,j-1} + \rho_{i-1,j-1})).$$

The combined effect is that Δp is approximated by the skewed Laplacian:

$$\Delta_h^{\mathrm{sk}} p = \frac{1}{2h^2}(p_{i-1,j-1} + p_{i+1,j-1} + p_{i+1,j+1} + p_{i-1,j+1} - 4p_{i,j}),$$

where we have taken $\Delta x = \Delta y \equiv h$. As is well known, this operator has the bad property that it leads to two decoupled grids, and the solution on one grid can be quite different from the solution on the other.

If one considers $\Delta_h^{9k} p = f$ on the unit square and attempts to solve it by relaxation, a local mode analysis of Fourier components of the error shows that the component that oscillates in both the x- and y-directions with a half-wavelength equal to the mesh spacing is not smoothed at all. This coasting or hourglass instability is common to Lagrangian codes. From the point of view of relaxation on a single grid, this is not serious since this frequency is inherently present in the solution—another way of saying that there are two completely decoupled grids; frequencies in the error near this frequency that should be smoothed are smoothed at the same slow rate as frequencies near the constant frequency. A basic premise of the multigrid method is that high-frequency components of the error can be smoothed efficiently on the fine grid. This premise is simply not true for the operator Δ_h^{9k}, and since frequencies near the highest one are introduced by interpolation, the multigrid method accentuates the problem for such differencing. If the grids remained decoupled as a function of time one could treat each separately by the multigrid method, but in Lagrangian codes the two grids become somewhat connected and yet there are still high frequencies that are not efficiently smoothed. A simple example is to take $\Delta y = 2\Delta x$, in which case the Laplacian is approximated by the nine-point stencil

$$\frac{1}{16\Delta x^2}\begin{bmatrix} 5 & -6 & 5 \\ 6 & -20 & 6 \\ 5 & -6 & 5 \end{bmatrix};$$

the high-frequency component is not smoothed for this operator.

As is well known [7], the skewed Laplacian also arises when piecewise continuous bilinear elements are used on a rectangular grid if only one Gauss point is used to compute the quadratures. And indeed when some of the finite-element experiments were repeated with only one Gauss point, it was found that the multigrid method converges no more rapidly than SOR.

One possible remedy is to change the differencing in SALE. This hourglass mode is usually smoothed away in Lagrangian codes in some ad hoc fashion. To avoid such an ad hoc procedure, it is interesting to try an alternate differencing that uses the standard five-point Laplacian and hence couples the grids together. In SALE the volume changes in the pressure iteration are calculated by forming $\Delta t(\nabla \cdot \mathbf{u}^{(k)})$. By exploiting the relationship between u, v, and p, one may write this as $\Delta t^2 \nabla \cdot ((1/\rho) \nabla p^{(k)}) + \Delta t \nabla \cdot {}^n\mathbf{u}$. One can then difference $\nabla \cdot ((1/\rho) \nabla p)$ by the standard five-point difference operator; however, as the mesh distorts, mixed derivatives enter, and the differencing becomes a nine-point scheme. When the pressure iteration has converged to a given tolerance, the new velocities are calculated by a discrete version of

$$\mathbf{u}^{n+1} = \mathbf{u}^n - \frac{\Delta t \, \nabla \mathbf{p}^{n+1}}{\rho}.$$

Since the velocities are updated only once, after the pressure iteration has converged, this approach has the advantage that it requires considerably less computer time for each iteration.

When no Lagrangian interfaces are present, so that the material density varies smoothy, the above method performs well. When interfaces are present, we found it necessary to take smaller time steps with this approach than with the skewed Laplacian. For a 12 $\times$ 12 grid, for example, and our sample problem below, the new method required a time step approximately one-half the time step of the previous approach. In this problem it appears that the standard five-point operator also has a slightly larger truncation error than the skewed Laplacian. For a very low speed flow and a sufficiently large density discontinuity at the material interface this affects the accuracy of the calculation and necessitates the smaller time step. As the grid is refined the difference in truncation error between the two methods decreases, and they approach one another. Tests on a 24 $\times$ 24-cell mesh have verified this thesis.

Let us examine the efficiency of the multigrid approach for the new differencing by comparing calculational time as required by multigrid with that required when all iterations are done on a fine grid. In this comparison we will apply multigrid in the mode that solves for the pressure field first and then upon convergence updates the velocity field as outlined above. The test problem we choose is the Rayleigh–Taylor instability for almost incompressible flow with a two-to-one density jump at the interface.

Initially the velocity field at the interface is perturbed with the function $v = \cos(\pi x/40)$. We use a stiffened gas equation of state, $p = a^2(\rho - \rho_0)$ with $a^2 = 2 \times 10^4$. The Mach number of this problem is of order 5×10^{-4}. Although the problem would run more smoothly if the continuous rezone option were involved, we ran the problem in the Lagrangian mode to test the effects of mesh distortion on the multigrid method. At time $t = 25$, the mesh distorts sufficiently that the time step shrinks and the problem must be terminated. The final mesh configuration appears in Fig. 6.

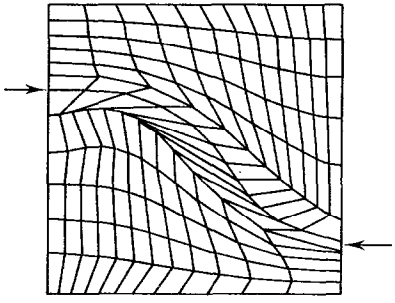

FIG. 6. Lagrangian mesh at end of calculation.

In Table II we give some time comparisons for this problem. We abbreviate multigrid by MG and single grid by SG. Normally the time step in SALE is changed dynamically. To make the comparison we ran with a piecewise constant time step. That is, the time step was halved at times near $t = 15.0$, 19.0, and 23.0. Given the relative constancy of the ratios in columns 3 and 4 of Table II for the 12 $\times$ 12 problem,

we felt justified in timing the 24×24 problem (which would have taken at least 8 times longer to run) only as far as $t = 3.0$. It is of interest to note that the improvement in running time is markedly increased as the mesh is refined. This is typical of multigrid.

TABLE II

Number of cells and time	Number of grids used (multigrid)	Total time spent iterating (MG /SG)	Total calculational time (MG /SG)	Fraction of calculation spent iterating (SG)	Fraction of calculation spent iterating (MG)
12×12 $t = 25.0$	3	0.67	0.75	0.81	0.73
12×12 $t = 24.0$	3	0.69	0.75	0.80	0.72
12×12 $t = 23.0$	3	0.67	0.75	0.79	0.75
12×12 $t = 19.0$	3	0.72	0.79	0.76	0.69
12×12 $t = 15.0$	3	0.69	0.77	0.73	0.70
12×12 $t = 3.0$	3	0.62	0.71	0.64	0.57
24×24 $t = 3.0$	4	0.39	0.51	0.80	0.60

As in the SOLA-ICE calculation, a single ω, this time 1.7, seemed to be nearly optimal for both single-grid calculations. We point out again that the single-grid calculation required the addition of the constant routine to be at all competitive with the multigrid approach. The success of the multigrid method in the context of Lagrangian hydrodynamics clearly indicates that such an approach would also perform well for implicit solutions of the diffusion equation

$$\frac{\partial u}{\partial t} - \nabla \cdot (D \nabla u) = f$$

on distorted grids.

V. Conclusion

It has been demonstrated that the multigrid method can be successfully applied to perform the pressure iteration in the Euelerian codes SOLA and SOLA-ICE. In both of these codes care was required in the interpolation routine to preserve certain functional relationships between the pressure and velocity fields. In SOLA-ICE there was the additional difficulty that in the limit of low speed flow, the basic relaxation scheme was inefficient; this inefficiency was traced to the slow convergence of the constant part of the solution and was remedied by the determination of a constant to add to the pressure field to accelerate the convergence of the constant part of the solution. We emphasize that the determination of a constant is useful not only in a multigrid context but also in the context of a relaxation on a single grid.

In addition to the above difficulties, the application of the multigrid method to a Lagrangian code had the difficulty that the difference equation for the pressure field had nonphysical high-frequency components in its solution (the standard hourglass instability). These components are inefficiently smoothed by the basic relaxation method; hence, the multigrid method is not successful. If, however, one changes the difference scheme—a change which can be argued on the grounds of avoiding nonphysical oscillations in the computed solution—then the multigrid method can also be applied to this Lagrangian code. The other interesting feature of the application to a Lagrangian code is that one can view the equation to be solved on the Lagrangian mesh as an equation with varying coefficients to be solved on a logical mesh. With proper averaging this is correct, and the logic of the multigrid method is no more complicated than it is for a rectangular mesh.

Appendix

In this appendix we show that (2.6) is actually equivalent to the traditional SOR method. Assume for simplicity that $\Delta x = \Delta y = h$. Then it is not hard to see that

$$\begin{aligned}\delta p_{i,j}^{(k+1)} = \frac{\omega\,\Delta x}{4\Delta t}\Big(u_{i-1,j}^{(k+1)} - u_{i,j}^{(k)} + \frac{\Delta t}{\Delta x}\,(\delta p_{i-1,j}^{(k+1)} + \delta p_{i+1,j}^{(k)}) \\ + v_{i,j-1}^{(k+1)} - v_{i,j}^{(k)} + \frac{\Delta t}{\Delta x}\,(\delta p_{i,j-1}^{(k+1)} + \delta p_{i,j+1}^{(k)})\Big) - \frac{\omega}{2}\,\delta p_{i,j}^{(k)}, \\ 2 \leqslant i \leqslant IM1, \quad 2 \leqslant j \leqslant JM1, \quad k \geqslant 0. \qquad \text{(A.1)}\end{aligned}$$

Also

$$\begin{aligned}\delta p_{i,j}^{(k)} &= \frac{\omega\,\Delta x}{4\Delta t}\Big(u_{i-1,j}^{(k+1)} - u_{i,j}^{(k)} + \frac{\Delta t}{\Delta x}\,\delta p_{i,j}^{(k)} + v_{i,j-1}^{(k+1)} - v_{i,j}^{(k)} + \frac{\Delta t}{\Delta x}\,\delta p_{i,j}^{(k)}\Big) \\ &= \frac{\omega}{2}\,\delta p_{i,j}^{(k)} + \frac{\omega\,\Delta x}{4\Delta t}\,(u_{i-1,j}^{(k+1)} - u_{i,j}^{(k} + v_{i,j-1}^{(k+1)} - v_{i,j}^{(k)}), \\ &\quad 2 \leqslant i \leqslant IM1, \quad 2 \leqslant j \leqslant JM1, \quad k \geqslant 0. \qquad \text{(A.2)}\end{aligned}$$

Solving for the divergence from Eq. (A.2) and substituting it into Eq. (A.1) yields

$$\delta p_{i,j}^{(k+1)} = (1 - \omega)\, \delta p_{i,j}^{(k)} + \frac{\omega}{4} (\delta p_{i-1,j}^{(k+1)} + \delta p_{i,j-1}^{(k+1)} + \delta p_{i-1,j}^{(k)} + \delta p_{i,j+1}^{(k)}),$$

$$2 \leqslant i \leqslant IM1, \quad 2 \leqslant j \leqslant JM1, \quad k \geqslant 0. \tag{A.3}$$

We may write Eq. (A.3) as

$$(D - \omega E)\, \delta p^{(k+1)} = ((1 - \omega)\, D + \omega F)\, \delta p^{(k)},$$

where D, E, and F are the appropriate diagonal, lower triangular, and upper triangular matrices as in [10, p. 58]. Now

$$(D - \omega E)\, p^0 = ((1 - \omega)\, D + \omega F)\, p^0 + \omega(\Delta x)^2\, \Delta_h p^0,$$

where $(1/(\Delta x)^2)\, \Delta_h$ is the usual five-point discrete Laplacian; thus

$$(D - \omega E)\, p^1 = ((1 - \omega)\, D + \omega F)\, p^0 + \omega(\Delta x)^2\, \Delta_h p^0 + D - \omega E)\, \delta p^0.$$

From Eq. (2.6),

$$(D - \omega E)\, \delta p_{i,j}^0 = \frac{\omega\, \Delta x}{\Delta t} (-u_{i,j}^0 + u_{i-1,j}^0 - v_{i,j}^0 + v_{i,j-1}^0),$$

so that

$$(D - \omega E)\, p^1 = ((1 - \omega)\, D + \omega F)\, p^0 + \omega c;$$

consequently

$$(D - \omega E)\, p^{(k+1)} = ((1 - \omega)\, D + \omega F)\, p^{(k)} + \omega c,$$

where

$$c_{i,j} = \frac{1}{\Delta t\, \Delta x} (a_{i,j} - a_{i-1,j} + b_{i,j} - b_{i,j-1}).$$

That is, Eq. (2.6) is equivalent to the traditional SOR iterative method [11] for the equation

$$p = \frac{1}{\Delta t} (c_x + c_y).$$

REFERENCES

1. A. A. AMSDEN AND C. W. HIRT, "YAQUI: An Arbitrary Lagrangian–Eulerian Computer Program For Fluid Flows at All Speeds," Los Alamos Scientific Laboratory Report No. LA-5100, 1973.
2. A BRANDT, *Math. Comp.* **31** (1977), 333–390.

3. JOHN R. RICE (Ed.), "Multi-Level Adaptive Techniques for Partial Differential Equations: Ideas and Software, Mathematical Software III," Proceedings of the Wisconsin Mathematics Software Conference, 1977.
4. L. D. CLOUTMAN, C. W. HIRT, AND N. C. ROMERO, "SOLA-ICE: A Numerical Algorithm for Transient Compressible Fluid Flows," Los Alamos Scientific Laboratory Report No. LA-6236, 1976.
5. J. E. DENDY, JR., Unpublished numerical experiments with the multi-grid method in a finite element code written by Charles Anderson.
6. G. E. FORSYTHE AND W. R. WASOW, "Finite-Difference Methods for Partial Differential Equations," Wiley, New York, 1960.
7. V. GIRAULT, *SIAM Numer. Anal.* **11** (1974), 260–282.
8. C. W. HIRT, B D. NICHOLS, AND N. C. ROMERO, "SOLA: A Numerical Solution Algorithm for Transient Fluid Flows," Los Alamos Scientific Laboratory Report No. LA-5852, 1975.
9. F. H. HARLOW AND A. A. AMSDEN, *J. Comp. Phys.* **8** (1971), 197.
10. R. S. VARGA, "Matrix Iterative Analysis," Prentice–Hall, Englewood Cliffs, N. J., 1962.
11. J. A. VIECELLI, *J. Comp. Phys.* **8** (1971), 119–143.