JOURNAL OF COMPUTATIONAL PHYSICS 113, 13–25 (1994)

A Fourth-Order Accurate Method for the Incompressible Navier–Stokes Equations on Overlapping Grids

WILLIAM D. HENSHAW*

IBM Research Division, Thomas J. Watson Research Centre, Yorktown Heights, New York, 10598

Received December 29, 1992; revised November 18, 1993

A method is described to solve the time-dependent incompressible Navier–Stokes equations with finite differences on curvilinear overlapping grids in two or three space dimensions. The scheme is fourth-order accurate in space and uses the momentum equations for the velocity coupled to a Poisson equation for the pressure. The boundary condition for the pressure is taken as $\nabla \cdot \mathbf{u} = 0$. Extra numerical boundary conditions are chosen to make the scheme accurate and stable. The velocity is advanced explicitly in time; any standard time stepping scheme such as Runge–Kutta can be used. The Poisson equation is solved using direct or iterative sparse matrix solvers or by the multigrid algorithm. Computational results in two and three space dimensions are given.

1. INTRODUCTION

A scheme is described for the accurate solution of the incompressible Navier–Stokes equations on regions with complicated geometry in two and three space dimensions. The approach uses fourth-order accurate finite differences on curvilinear overlapping grids. The momentum equations for the velocity, $\mathbf{u}$, are solved together with the Poisson equation for the pressure, p. One key element of the method is the choice of boundary conditions. A boundary condition for the pressure is required as well as extra numerical boundary conditions for the fourth-order method. The boundary condition for the pressure is taken as $\nabla \cdot \mathbf{u} = 0$. It will be argued that this is the natural boundary condition for p. Numerical boundary conditions are chosen by applying the equations on the boundary and by setting the normal derivative of $\nabla \cdot \mathbf{u}$ equal to zero on the boundary. An overlapping grid also requires boundary conditions on those boundaries where one component grid overlaps another. Solution values at these points are obtained by interpolation. This is the standard approach and is described in more detail later in the paper.

The velocity is advanced explicitly in time with a method of lines approach. Any standard time-stepping scheme such as a Runge–Kutta or multistep method can be used. At each stage in the time step the velocity is first updated and then the pressure is found by solving the Poisson equation. The overall accuracy in time is equal to the accuracy of the chosen time-stepping algorithm. The fourth-order accurate approximation to the Poisson equation is solved either with a direct sparse matrix solver, an iterative sparse matrix solver or with the multigrid algorithm. Some care is required to solve the Poisson equation since the system is usually singular (the pressure is only determined up to a constant). Numerical results in two and three space dimensions show the accuracy of the scheme. The stability and accuracy of the scheme described in this paper are analysed in [18], where a general principle for deriving numerical boundary conditions is also presented.

1.1. *Background*

The initial boundary-value problem for the incompressible Navier–Stokes equations is

$$\left.\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu \Delta \mathbf{u} - \mathbf{f} = \mathbf{0} \\ \nabla \cdot \mathbf{u} = 0 \end{aligned}\right\} \quad \mathbf{x} \in \Omega \qquad (1)$$

$$B(\mathbf{u}, p) = \mathbf{0} \quad \mathbf{x} \in \partial\Omega$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}) \quad \text{at} \quad t = 0.$$

Here p is the pressure and ν the kinematic viscosity, $\nu > 0$. The domain Ω lies in $\mathbf{R}^{n_d}$, where n_d, the number of space dimensions, is two or three. There are n_d boundary conditions denoted by $B(\mathbf{u}, p) = \mathbf{0}$. On a fixed wall, for example, the boundary conditions are $\mathbf{u} = \mathbf{0}$. System (1) will be called the velocity-divergence form of the equations.

There are alternative formulations to the velocity-divergence equations (1). By taking the curl of the momentum equations, a system for the vorticity $\omega = \nabla \times \mathbf{u}$ and velocity results which does not depend on the pressure. In two space dimensions this system is particularly convenient as it can be written in terms of two scalar functions, a stream function ψ, and the one nontrivial component of the

*Present address: Computing, Information and Communications Division, Los Alamos National Laboratory, Los Alamos, New Mexico, 87545. E-mail address: henshaw@c3.lanl.gov.

0021-9991/94 $6.00

vorticity. In three space dimensions many of the advantages of the vorticity formulation are lost and it becomes more attractive (in applying boundary conditions for example), to use velocity and pressure variables.

An alternative initial-boundary value problem, called the velocity–pressure formulation, is

$$\left.\begin{aligned} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p - \nu\,\Delta\mathbf{u} - \mathbf{f} = 0 \\ \Delta p + \nabla u\cdot\mathbf{u}_x + \nabla v\cdot\mathbf{u}_y + \nabla w\cdot\mathbf{u}_z - \nabla\cdot\mathbf{f} = 0 \end{aligned}\right\}, \qquad \mathbf{x}\in\Omega \quad (2)$$

$$\left.\begin{aligned} B(\mathbf{u}, p) = 0 \\ \nabla\cdot\mathbf{u} = 0 \end{aligned}\right\}, \qquad \mathbf{x}\in\partial\Omega$$

$$\mathbf{u}(\mathbf{x}, 0) = \mathbf{u}_0(\mathbf{x}) \qquad \text{at} \quad t = 0.$$

This is the form of the equations that will be discretized in the method described in this paper. The pressure equation is derived by taking the divergence of the momentum equation and using $\nabla\cdot\mathbf{u} = 0$. For this latter system an extra boundary condition is required in order to make the problem well-posed. The condition $\nabla\cdot\mathbf{u} = 0$ for $\mathbf{x}\in\partial\Omega$ is added as the extra boundary condition. Further remarks on this choice are given later.

Fourth-order accurate difference methods require extra (numerical) boundary conditions. The choice of these numerical boundary conditions is crucial to the creation of a stable and accurate scheme. For the scheme given here, boundary conditions are required to determine $\mathbf{u}$ and p at two lines of fictitious (ghost) points. Often, a useful technique for deriving boundary conditions for higher-order methods is to apply the equation (and its normal derivatives) on the boundary. Numerical boundary conditions are thus derived by applying the momentum equations and the pressure equation on the boundary. In addition the normal derivative of the divergence is specified on the boundary.

$$\text{Numerical BCs:} \quad \begin{cases} \mathbf{u}_t + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \nu\,\Delta\mathbf{u} + \mathbf{f} \\ \qquad = 0, \qquad \mathbf{x}\in\partial\Omega \\ \Delta p + \nabla u\cdot\mathbf{u}_x + \nabla v\cdot\mathbf{u}_y + \nabla w\cdot\mathbf{u}_z - \nabla\cdot\mathbf{f} \\ \qquad = 0, \qquad \mathbf{x}\in\partial\Omega \\ (\partial/\partial n)(\nabla\cdot\mathbf{u}) = 0, \qquad \mathbf{x}\in\partial\Omega. \end{cases} \quad (3)$$

On the second fictitious line the pressure and the tangential components of the velocity are extrapolated. There is also a second-order accurate version of this method that does not require these extra numerical boundary conditions.

1.2. *Discussion*

It is appropriate, perhaps, to make some remarks regarding the choice of $\nabla\cdot\mathbf{u} = 0$ as the extra boundary condition for the velocity–pressure formulation. The extra boundary condition required by the velocity–pressure formulation should satisfy three conditions. (i) It should be chosen so that (2) (plus compatibility constraints) is well posed. (ii) It should be consistent with the original formulation (1). (iii) It should be chosen so that formulation (2) is equivalent to (1). These three conditions are satisfied by the boundary condition $\nabla\cdot\mathbf{u} = 0$, which, although it does not look like a pressure boundary condition, is in some sense the natural extra condition to add. It is not hard to show that (2) is equivalent to (1) at least for solutions that are sufficiently smooth.

The question of what boundary condition to use for the pressure equation has led to much discussion, see for example [11, 19]. These papers consider, for example, whether it is appropriate to use the tangential or normal component of the momentum equation on the boundary as a boundary condition for the pressure equation. Gresho and Sani [11] present an extended discussion of pressure boundary conditions where they conclude that the most appropriate condition is the use of the normal component of the momentum equation on the boundary. It appears, however, that essentially all methods also impose (implicitly or explicitly) $\nabla\cdot\mathbf{u} = 0$ on the boundary. Often the fact that this condition is applied is not emphasized. On staggered grids, for example, the condition occurs naturally. In related work Karniadakis *et al.* [19] consider boundary conditions for a projection method for the spectral element method. They use the normal component of the momentum equation but find that in this boundary condition they must write $\Delta\mathbf{u}$ as $\nabla(\nabla\cdot\mathbf{u}) - \nabla\times(\nabla\times\mathbf{u})$ and set $\nabla(\nabla\cdot\mathbf{u}) = 0$.

It seems that much of the confusion goes away if one realizes that the essential boundary condition for the pressure equation is $\nabla\cdot\mathbf{u} = 0$. This is the natural condition to add so that the velocity–pressure formulation is equivalent to the velocity-divergence formulation. The velocity–pressure formulation is **not** equivalent to the velocity-divergence formulation if the extra boundary condition is instead taken as the normal component of the momentum equation for there is nothing to stop the divergence from becoming nonzero in the domain if it is nonzero on the boundary. Indeed this later boundary condition alone adds no new information and would lead to an under-determined system, as pointed out by Strikwerda [24]. However, from a discrete point of view it seems appropriate to apply the momentum equation on the boundary as an extra numerical boundary condition. This boundary condition, combined with the (essential) boundary condition $\nabla\cdot\mathbf{u} = 0$ then leads to a boundary condition on the normal derivative of the pressure that can be conveniently used when solving the discrete Poisson equation for the pressure.

The advantages of using higher-order accurate methods for the solution of the incompressible Navier–Stokes equa-

tions (or many other PDEs) are by now well known. On periodic regions, Fourier spectral methods are now widely used. Fourth-order methods, although not as accurate as spectral methods, still offer significant advantages over second-order methods. Applying a high-order method on a complicated region is a difficult task. One major difficulty is in the generation of a grid for the region. The grid must be smooth enough so that errors associated with variations in the grid do not overwhelm the errors in the method. A second difficulty is developing a high-order method that works in general regions. There are a number of approaches to generating grids for complicated geometries. These approaches include multiblock patched grids [25], unstructured triangular (tetrahedral) grids [2] and overlapping (overlaid) grids [23, 8]. With multiblock patched grids one can use higher-order finite-difference methods or the spectral element method [19]. Higher order finite element or finite-volume methods can be used on unstructured grids. Each grid generation approach has its own advantages and disadvantages. Unstructured grids are very flexible but they are not as efficient as structured grids; moreover, the problem of creating a good triangular grid to resolve viscous boundary layers is still an open question. Multiblock patched grids are efficient but less flexible—the problem of automatically dividing a three-dimensional region into blocks is still a problem. Overlapping grids represent a compromise; they are more flexible than patched grids but are still highly structured.

In this paper an approach is presented that uses overlapping grids and high-order finite differences. Overlapping grids can be used to create smooth grids on complicated regions. The grid construction program CMPGRD is used to create overlapping grids in two and three space dimensions [8]. An overlapping grid consists of a set of component grids which cover a region and overlap where they meet. Interpolation is used to match the solution between component grids. As each component grid is logically rectangular it is straight forward to apply high-order finite difference methods in an efficient manner. Overlapping grids can be created that are free from artificial coordinate singularities. For example, the singularities at the poles of a sphere associated with the standard spherical polar coordinates can be avoided by using two (or more) overlapping patches to cover the sphere [17]. A less severe form of singularity appears in multiblock patched grids where the metric coefficients are sometimes discontinuous across block boundaries. Overlapping grids can avoid these types of singularities, thus making it much easier to obtain high-order accuracy.

High-order accurate methods on overlapping grids have be used successfully for a variety of problems. In [6], for example, Browning has used fourth- and six-order methods on overlapping grids to solve the shallow water equations on a sphere. In [8, 28] fourth-order accurate methods are used to solve elliptic problems and nonlinear eigenvalue problems on overlapping grids.

In related work, the three-dimensional incompressible Navier–Stokes equations have been solved on overlapping grids by Kiris *et al.* [21]. They use the artificial compressibility method and second-order accurate differences to compute the flow in an artificial heart. These computations show some of the advantages of using overlapping grids for computing flows with moving boundaries in complex three-dimensional regions. Tu and Fuchs also solve the three-dimensional incompressible equations on overlapping grids for flows in internal combustion engines. They use a second-order accurate method coupled with a multigrid algorithm [26]. Recently Wright and Shyy [29] have described a method for multi-block grids where they are concerned with conservation at the interface boundaries. The approach described in this paper has similarities to the well-known projection method, originally developed by Chorin [9]. See, for example, the article by Bell, Colella, and Glaz [3] for a discussion of projection methods.

2. SPATIAL DISCRETIZATION

The equations defining the velocity–pressure formulation are discretized in space using finite-difference methods on overlapping grids. An overlapping grid consists of a set of logically rectangular grids that cover a region and overlap where they meet. Interpolation conditions are used to connect the solutions on different component grids. Associated with each component grid (numbered $k = 1, 2, ..., n_g$) there is a transformation, $\mathbf{d}_k$, that maps the unit cube, with coordinates denoted by $\mathbf{r} = (r_1, r_2, r_3)$, into physical space, $\mathbf{x} = (x_1, x_2, x_3)$,

$$\mathbf{x}(\mathbf{r}) = \mathbf{d}_k(\mathbf{r}).$$

Each component grid, G_k, consists of a set of grid points,

$$G_k = \{\mathbf{x}_{i,k} \mid i = (i_1, i_2, i_3),\ n_{m,a,k} - 2 \leqslant i_m \leqslant n_{m,b,k} + 2,\ m = 1, 2, 3\}.$$

Two extra lines of fictitious points are added for convenience in discretizing to fourth order. Boundaries of the computational domain will coincide with the boundaries of the unit cubes, $i_m = n_{m,a,k}$ or $i_m = n_{m,b,k}$ for $m = 1, ..., n_d$. Henceforth, the subscript k, denoting the component grid, will normally not be written.

Let $\mathbf{U}_i$ and P_i denote the discrete approximations to $\mathbf{u}$ and p so that

$$\mathbf{U}_i \approx \mathbf{u}(\mathbf{x}_i), \qquad P_i \approx p(\mathbf{x}_i).$$

Here $\mathbf{U}_i = (U_{1i}, U_{2i}, U_{3i})$ is the vector containing the

cartesian components of the velocity. The momentum and pressure equations are discretized with fourth-order accurate central differences applied to the equations written in the unit cube coordinates, as will now be outlined. Define the shift operators E_{+m} and E_{-m} in the coordinate direction m by

$$E_{\pm m}\mathbf{U}_i = \begin{cases} \mathbf{U}_{i_1 \pm 1, i_2, i_3} & \text{if } m=1 \\ \mathbf{U}_{i_1, i_2 \pm 1, i_3} & \text{if } m=2 \\ \mathbf{U}_{i_1, i_2, i_3 \pm 1} & \text{if } m=3 \end{cases} \tag{4}$$

and the difference operators

$$D_{\pm m} = E_{\pm m} - 1$$
$$D_{\pm m_1, \pm m_2} = E_{\pm m_1} E_{\pm m_2} - 1.$$

Let D_{4r_m}, $D_{4r_m r_n}$, $D_{4x_m x_n}$ denote fourth-order accurate derivatives with respect to $\mathbf{r}$ and $\mathbf{x}$. The derivatives with respect to $\mathbf{r}$ are the standard fourth-order centred difference approximations. For example,

$$\frac{\partial \mathbf{u}}{\partial r_m} \approx D_{4r_m}\mathbf{U}_i := \frac{(-E_{+m}^2 + 8E_{+m} - 8E_{-m} + E_{-m}^2)\,\mathbf{U}_i}{12(\Delta r_m)}$$

$$\frac{\partial^2 \mathbf{u}}{\partial r_m^2} \approx D_{4r_m r_m}\mathbf{U}_i$$
$$:= \frac{(-E_{+m}^2 + 16E_{+m} - 30 + 16E_{-m} - E_{-m}^2)\,\mathbf{U}_i}{24(\Delta r_m)^2},$$

where $\Delta r_m = 1/(n_{m,b} - n_{m,a})$. The derivatives with respect to $\mathbf{x}$ are defined by the chain rule,

$$\frac{\partial \mathbf{u}}{\partial x_m} = \sum_n \frac{\partial r_n}{\partial x_m}\frac{\partial \mathbf{u}}{\partial r_n} \approx D_{4x_m}\mathbf{U}_i := \sum_n \frac{\partial r_n}{\partial x_m} D_{4r_n}\mathbf{U}_i$$

$$\frac{\partial^2 \mathbf{u}}{\partial x_m^2} = \sum_{n,l} \frac{\partial r_n}{\partial x_m}\frac{\partial r_l}{\partial x_m}\frac{\partial^2 \mathbf{u}}{\partial r_n r_l} + \sum_n \frac{\partial^2 r_n}{\partial x_m^2}\frac{\partial \mathbf{u}}{\partial r_n}$$

$$\approx D_{4x_m x_m}\mathbf{U}_i := \sum_{n,l} \frac{\partial r_n}{\partial x_m}\frac{\partial r_l}{\partial x_m} D_{4r_n r_l}\mathbf{U}_i$$

$$+ \sum_n \left(D_{4x_m}\frac{\partial r_n}{\partial x_m} \right) D_{4r_n}\mathbf{U}_i.$$

The entries in the Jacobian matrix, $\partial r_m/\partial x_n$, are assumed to be known at the vertices of the grid; these values are obtained from the grid generation program, CMPGRD. The spatial discretizations of the momentum and pressure equations can thus be written as

$$\frac{d}{dt}\mathbf{U}_i + (\mathbf{U}_i \cdot \nabla_4)\,\mathbf{U}_i + \nabla_4 P_i - \nu\,\Delta_4 \mathbf{U}_i - \mathbf{f}_i = 0$$

$$\Delta_4 P_i + \sum_{m=1}^{n_d} \nabla_4 U_{m,i} \cdot D_{4x_m}\mathbf{U}_i - \nabla_4 \cdot \mathbf{f}_i = 0,$$

where

$$\nabla_4 U_i = (D_{4x_1} U_i, D_{4x_2} U_i, D_{4x_3} U_i)$$
$$\nabla_4 \cdot \mathbf{U}_i = D_{4x_1} U_{1,i} + D_{4x_2} U_{2,i} + D_{4x_3} U_{3,i}$$
$$\Delta_4 \mathbf{U}_i = (D_{4x_1 x_1} + D_{4x_2 x_2} + D_{4x_3 x_3})\,\mathbf{U}_i.$$

In order to help keep the discrete divergence small, an additional term proportional to $\nabla_4 \cdot U$ will be added to the pressure equation. This will be described later in the paper. It can be seen that the discretization of the equations is acomplished in a straightforward manner with overlapping grids. However, the remaining steps of setting up the matrix for the pressure equation and solving the boundary conditions are more difficult.

Discretizing the boundary conditions. For the purposes of this discussion assume that the boundary condition for $\mathbf{u}$ is of the form $\mathbf{u}(\mathbf{x}, t) = \mathbf{u}_B(\mathbf{x}, t)$ for $\mathbf{x} \in \partial\Omega$. More general boundary conditions on $\mathbf{u}$ and p, such as extrapolation conditions, can also be dealt with, although some of the details of implementation may vary. At a boundary the conditions

$$\left.\begin{aligned} \mathbf{U}_i - \mathbf{u}_B(\mathbf{x}_i) &= 0 \\ \nabla_4 \cdot \mathbf{U}_i &= 0 \\ D_{4n}(\nabla_4 \cdot \mathbf{U}_i) &= 0 \\ \frac{d}{dt}\mathbf{U}_i + (\mathbf{U}_i \cdot \nabla_4)\,\mathbf{U}_i & \\ + \nabla_4 P_i - \nu\,\Delta_4 \mathbf{U}_i - \mathbf{f}_i &= 0 \\ \Delta_4 P_i + \sum_{m=1}^{n_d} \nabla_4 U_{m,i} \cdot D_{4x_m}\mathbf{U}_i & \\ - \nabla_4 \cdot \mathbf{f}_i &= 0 \end{aligned}\right\} \text{ for } i \in \text{boundary}$$

$$\left.\begin{aligned} \mathbf{t}_\mu \cdot D_{+m}^4 \mathbf{U}_i &= 0 \\ D_{+m}^4 P_i &= 0 \end{aligned}\right\} \text{ for } i \in \text{second fictitious line}$$

are applied, where $\mathbf{t}_\mu$, $\mu = 1, n_d - 1$, are linearly independent vectors that are tangent to the boundary. In the extrapolation conditions either D_{+m} or D_{-m} should be chosen, as appropriate. Thus at each point along the boundary there are 12 equations for the 12 unknowns $(\mathbf{U}_i, P_i)$ located on the boundary and the two lines of fictitious points. Note that two of the numerical boundary conditions couple the pressure and velocity. In order to advance the velocity with an explicit time stepping method it is convenient to decouple the solution of the pressure equation from the solution of the velocity. A procedure to accomplish this is described in the next section on time stepping.

3. TIME STEPPING

A method of lines approach is used to solve the equations in time. After discretizing the equations in space, one

can regard the result as a system of ordinary differential equations,

$$\frac{d}{dt}\mathbf{U} = F(\mathbf{U}, t), \tag{5}$$

where the pressure is considered to be a function of the velocity, $P = P(\mathbf{U})$. The equations are integrated using an explicit time-stepping scheme (implicit methods, such as the fractional-step methods [20, 19], could also be used). Typical explicit time-stepping procedures such as Runge–Kutta methods or multistep methods advance the solution in time by solving one or more substeps of the form

$$\mathbf{U}(t) = \mathbf{U}(*) + \alpha\, \Delta t\, F(\mathbf{U}(t-\Delta t), t-\Delta t), \tag{6}$$

where $\mathbf{U}(*)$ is some known function, depending on the solution at previous time steps. Assume that at time $t-\Delta t$ the values for $\mathbf{U}(t-\Delta t)$ and $P(t-\Delta t)$ are known at all nodes (interior, boundary, and fictitious nodes) and that the values of $F(\mathbf{U}(t-\Delta t), t-\Delta t)$ are known at all interior nodes. Here are the steps to advance one substep and determine these values at time t:

Step 1. Determine $\mathbf{U}(t)$ at all interior nodes using the substep formula (6).

Step 2. Determine the velocity $\mathbf{U}(t)$ at all boundary and fictitious points by solving the boundary conditions

$$\left.\begin{aligned} \mathbf{U}_i(t) - \mathbf{u}_B(\mathbf{x}_i, t) &= 0 \\ \nabla_4 \cdot \mathbf{U}_i(t) &= 0 \\ D_{4n}(\nabla_4 \cdot \mathbf{U}_i(t)) &= 0 \\ \mathbf{t}_\mu \cdot \left\{ \frac{d}{dt}\mathbf{U}_i(t) + (\mathbf{U}_i(t)\cdot\nabla_4)\,\mathbf{U}_i(t) \right. & \\ \left. + \nabla_4 P_i^*(t) - \nu \nabla_4 \mathbf{U}_i(t) - \mathbf{f}_i \right\} &= 0 \end{aligned}\right\} \text{for } i \in \text{boundary} \tag{7}$$

$$\mathbf{t}_\mu \cdot D^4_{+m}(\mathbf{U}_i(t)) = 0 \quad \text{for } i \in \text{second fictitious line},$$

where $\mu = 1, n_d - 1$. Only the tangential components of the momentum equations are used in this step. The pressure at time t is not known yet but the numerical boundary condition requires knowledge of $\mathbf{t}_\mu \cdot \nabla_4 P(t)$. This quantity is approximated by extrapolating the pressure in time giving an approximate value denoted by $P^*(t)$. Since this extrapolated value for the pressure on the boundary is only used as a numerical boundary condition one can expect from the theory that the overall accuracy and stability of the method will not be affected [18]. At each point on the boundary, equation (7) gives nine equations for the nine unknown values of the velocity at the boundary and two fictitious points.

Step 3. Solve the pressure equation with the remaining boundary conditions,

$$\Delta_4 P_i(t) = -\sum_{m=1}^{n_d} \nabla_4 U_{m,i} \cdot D_{4x_m} \mathbf{U}_i + \nabla_4 \cdot \mathbf{f}_i(t),$$

$$i \in \text{interior and boundary points}$$

$$\mathbf{n} \cdot \nabla_4 P_i(t) = -\mathbf{n} \cdot \left(\frac{\partial \mathbf{U}_i(t)}{\partial t} + (\mathbf{U}_i(t)\cdot\nabla_4)\,\mathbf{U}_i(t) - \nu\, \Delta_4 \mathbf{U}_i(t) - \mathbf{f}(t) \right), \quad i \in \text{boundary points} \tag{8}$$

$$D^4_{+m} P_i(t) = 0, \quad i \in \text{second fictitious line}.$$

The velocity, $\mathbf{U}(t)$, is known at all points and thus the right-hand sides in (8) can be determined. The numerical boundary condition that came from the normal derivative of the momentum equation appears here as a Neumann boundary condition for the pressure.

Step 4. Given the pressure, $P_i(t)$ its gradient can be computed at all interior points and thus $F(\mathbf{U}(t), t)$ can be determined at all interior nodes. This completes the time step.

In summary:

0. Given $\mathbf{U}(t-\Delta t)$ and $P(t-\Delta t)$ at all nodes and $F(\mathbf{U}(t-\Delta t), t-\Delta t)$ at interior nodes:

1. Compute $\mathbf{U}(t)$ at all interior nodes by making a time step.

2. Compute $\mathbf{U}(t)$ at boundary and fictitious points using some of the boundary conditions

3. Compute the right-hand side to the pressure equation and the right-hand side to the pressure boundary condition and then solve for pressure everywhere, $P(t)$.

4. Compute $F(\mathbf{U}(t), t)$ at interior nodes.

Solving the Pressure Equation. The discretized pressure equation takes the form of a linear system of equations:

$$A\mathbf{P} = \mathbf{F} \tag{9}$$

Here $\mathbf{P}$ is the vector containing the values of P_i at all grid points. Normally the matrix A will be singular because the pressure is only known up to an arbitrary constant. (This may not be true if an inflow or outflow boundary condition sets the level of the pressure). When A is singular, the discrete system will only have a solution if $\mathbf{F}$ is in the column space of A, or, equivalently, if $\mathbf{F}$ is orthogonal to the null-space of A^t. This compatibility condition is the discrete analogue of the solvability condition for Laplace's equation

with Neumann boundary conditions which states that a necessary condition for

$$\begin{aligned} \Delta p &= f, \qquad \mathbf{x} \in \Omega, \\ \frac{\partial}{\partial n} p &= g, \qquad \mathbf{x} \in \partial\Omega, \end{aligned} \tag{10}$$

to have a solution is that

$$\int_{\Omega} f \, d\mathbf{x} = \int_{\partial\Omega} g \, ds.$$

With some numerical methods the discrete compatibility condition is automatically satisfied. This is true, for example, with methods using staggered grids. Abdallah [1] shows how to design a scheme on nonstaggered grids so that the discrete compatibility condition is exactly satisfied. In the approach presented here the discrete compatibility condition is not exactly satisfied (although it should be satisfied to fourth-order accuracy). It has been found that a good approach to solving (9) is to solve the augmented system

$$\begin{bmatrix} A & \mathbf{r} \\ \mathbf{r}^t & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \alpha \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \beta \end{bmatrix}, \tag{11}$$

where $\mathbf{r} = [1, 1, \ldots, 1]^t$ is the right null vector of A. Although the matrix A is singular the above augmented system is nonsingular. If $\mathbf{l}$ denotes the left eigenvector of A (which never has to be calculated) then the solution to the augmented system satisfies

$$\alpha = \frac{\mathbf{l}^t \mathbf{F}}{\mathbf{l}^t \mathbf{r}} \tag{12}$$

$$A\mathbf{P} = \mathbf{F} - \alpha \mathbf{r} \tag{13}$$

$$\mathbf{r}^t \mathbf{P} = \beta. \tag{14}$$

Equation (13) has a solution because the right-hand side is orthogonal to $\mathbf{l}$. The general solution to (13) is equal to a particular solution plus an arbitrary constant times the right null vector. Equation (14) determines this arbitrary constant. From (12) it can be seen that α measures the degree to which the discrete compatibility condition is satisfied; one can expect that $\alpha = O(h^4)$ in the present case. The value chosen for β sets the arbitrary constant in the pressure.

The discrete pressure equation (11) is solved in various ways using sparse matrix solvers. Direct sparse solvers are used for small problems and iterative solvers, such as biconjugate gradient squared, GMRES, or multigrid [16, 14], are used for larger problems.

4. SOME DETAILS OF THE IMPLEMENTATION

There are a number of implementation details that must be considered when writing a general program for overlapping grids. These include, for example, the discretization of the equations near the boundaries (especially corners and edges), interpolation, and the initialization of the solution. In this section we discuss how some of these problems were addressed. Fortunately, since our grids are free of coordinate singularities, it is not necessary to deal with that difficulty.

4.1. *Interpolation Points*

The values on the boundary of a component grid, k_1, that overlaps a second component grid k_2 are obtained by interpolation. Interpolation is performed in the unit cube coordinates of grid k_2 using triquartic interpolation with a $5 \times 5 \times 5$ stencil of points. In Chesshire *et al.* [8] it was shown how to choose the order of accuracy of the interpolation formula to be consistent with the discretization of the PDE.

To be specific suppose that

$\mathbf{x}_{i_1,k_1}$ = point on grid k_1 to be interpolated from grid k_2

$\mathbf{r}' = \mathbf{d}_{k_2}^{-1}(\mathbf{x}_{i_1,k_1})$ = position of $\mathbf{x}_{i_1,k_1}$ in the unit cube of grid k_2

i_2, k_2 = index of the lower corner of the stencil to be used for interpolation

$\mathbf{r}(i_2, k_2)$ = unit square coordinates of the point (i_2, k_2).

Recall that $\mathbf{x} = \mathbf{d}_{k_2}(\mathbf{r})$ is the mapping from the unit cube onto the region covered by component grid k_2. The grid generation program CMPGRD supplies the coordinates $\mathbf{r}'$ of each point that needs to be interpolated. The interpolation is defined in terms of the Lagrange polynomials

$$q_i(s) = \frac{\prod_{j=0, j\neq i}^{4} (s-j)}{\prod_{j=0, j\neq i}^{4} (i-j)}.$$

Let $\mathbf{s} = (s_1, s_2, \ldots, s_{n_d})^{\mathrm{T}}$ be the normalized position of the point to be interpolated relative to the corner of the stencil of interpolation points:

$$s_m = \frac{r'_m - r_m(i_2, k_2)}{\Delta r_m}, \qquad m = 1, 2, \ldots, n_d.$$

The triquartic Lagrange interpolation formula is then

$$\mathbf{U}_{i_1,k_1} = \sum_{j_1=0}^{4} q_{j_1}(s_1) \sum_{j_2=0}^{4} q_{j_2}(s_2) \sum_{j_3=0}^{4} q_{j_3}(s_3)\, \mathbf{U}_{i_2+(j_1,j_2,j_3),k_2}.$$

A standard utility routine is used to perform the interpolation. The interpolation coefficients are stored and the interpolation can be vectorized on machines with gather-scattered operations.

4.2. *Edges and Vertices*

An important special case concerns obtaining solution values at points that lie near edges and vertices of grids (or corners of grids in 2D). Define a *boundary edge* to be the edge that is formed at the intersection of adjacent faces of the unit cube where both faces are boundaries of the computational domain. Along a boundary edge, values of the solution are required at the fictitious points in the region exterior to both boundary faces. For example, suppose that the edge defined by $i_1 = n_{1,a}$, $i_2 = n_{2,a}$, and $i_3 = n_{3,a}, \ldots, n_{3,b}$ is a boundary edge. Values must be determined at the exterior points $i = (n_{1,a}+m, n_{2,a}+n, i_3)$ for $m, n = -2, -1$. The following conditions are imposed:

$$\frac{\partial}{\partial r_m}(\nabla \cdot \mathbf{u}) = 0, \qquad m = 1, 2, \tag{15}$$

$$\mathbf{t}_3 \cdot D^6_{+,1,2}\mathbf{U}_{i_1-1,i_2-1,i_3} = 0. \tag{16}$$

Here $\mathbf{t}_3$ is the unit vector in the direction of the edge. Recall that $D_{+,1,2}\mathbf{U}_i = \mathbf{U}_{i_1+1,i_2+1,i_3} - \mathbf{U}_i$ and thus condition (16) is an extrapolation into the region, of the component of the velocity that is parallel to the edge. Equations (15), (16) supply sufficient information to determine the values of the fictitious points outside the edge, as will now be shown. By expanding $\mathbf{u}(-r_1, -r_2, r_3)$ and $\mathbf{u}(+r_1, +r_2, r_3)$ in a Taylor series about (0, 0, 0) it follows that

$$\begin{aligned}\mathbf{u}(-r_1, -r_2, r_3) &= 2\mathbf{u}(0,0,r_3) - \mathbf{u}(r_1, r_2, r_3) \\ &\quad + \tfrac{1}{2}r_1^2\mathbf{u}_{r_1r_1}(0,0,r_3) + r_1r_2\mathbf{u}_{r_1r_2}(0,0,r_3) \\ &\quad + \tfrac{1}{2}r_2^2\mathbf{u}_{r_2r_2}(0,0,r_3) + O(|r_1|^4 + |r_2|^4).\end{aligned} \tag{17}$$

The derivatives $\mathbf{u}_{r_1r_1}$ and $\mathbf{u}_{r_2r_2}$ are tangential derivatives (on the appropriate face) and can be computed from the given boundary data. Here it is assumed that the given boundary data are compatible at edges. The mixed derivative term, $\mathbf{u}_{r_1r_2}$, remains to be determined. When expanded by the chain rule equations (15) can be written as

$$\sum_{l,n=1}^{3} \frac{\partial r_n}{\partial x_l}\frac{\partial^2 u_l}{\partial r_m\, \partial r_n} + \sum_{l,n=1}^{3} \frac{\partial^2 r_n}{\partial r_m\, \partial x_l}\frac{\partial u_l}{\partial r_n} = 0$$

for $m = 1, 2$. The only term in these equations that is not known from the boundary data is the mixed derivative term $\mathbf{u}_{r_1r_2}$; and thus there are two equations for the three unknown components of $\mathbf{u}_{r_1r_2}$. To obtain a third equation for $\mathbf{u}_{r_1r_2}$ the extrapolation condition (16) is combined with the equation formed when the tangent vector $\mathbf{t}_3$ is dotted into (17) (with $r_1 = -\Delta r_1$, $r_2 = -\Delta r_2$). After solving for $\mathbf{u}_{r_1r_2}$, (17) gives a fourth-order accurate approximation to the four solution values that lie outside the boundary edge.

In two space dimensions, the values outside a corner are determined in a similar manner, although the extrapolation condition is not required.

At a vertex in 3D it follows from Taylor series that

$$\mathbf{u}(-\mathbf{r}) = 2\mathbf{u}(\mathbf{0}) - \mathbf{u}(\mathbf{r}) + \sum_{mn} r_m r_n \frac{\partial^2 \mathbf{u}}{\partial r_m\, \partial r_n}(\mathbf{0}) + O(|\mathbf{r}|^4).$$

All of the second-order derivatives $\mathbf{u}_{r_mr_n}$ are tangential derivatives on one of the faces that meets at the vertex and thus are known in terms of the given boundary values. Thus the value of $\mathbf{U}_i$ at the eight points which lie outside a vertex can be computed.

4.3. *Solving the Numerical Boundary Equations*

The numerical boundary conditions (7) define the values of $\mathbf{U}$ on two lines of fictitious points in terms of values of the velocity on the boundary and the interior. The equations couple the unknowns in the tangential direction to the boundary so that in principle a system of equations for all boundary points must be solved. However, when the grid is nearly orthogonal to the boundary there is a much more efficient way to solve the boundary conditions. The first step in the algorithm is to solve for the tangential components of the velocity from

$$\left.\begin{aligned} &\mathbf{U}_i(t) - \mathbf{u}_B(\mathbf{x}_i, t) = 0 \\ &\mathbf{t}_\mu \cdot \Big\{ \frac{d}{dt}\mathbf{U}_i(t) + (\mathbf{U}_i(t)\cdot\nabla_4)\,\mathbf{U}_i(t) \\ &\qquad + \nabla_4 P^*(t) - \nu\, \Delta_4 \mathbf{U}_i(t) - \mathbf{f} \Big\} = 0 \end{aligned}\right\} \quad \text{for } i \in \text{boundary}$$

$$\mathbf{t}_\mu \cdot D^6_{+m}(\mathbf{U}_i(t)) = 0 \qquad \text{for } i \in \text{second fictitious line.}$$

If the grid is orthogonal to the boundary then the discrete Laplacian applied at boundary will not have any mixed derivative terms. Therefore the only fictitious points appearing in the equation applied at the boundary point (i_1, i_2, i_3) will be the two points $(i_1, i_2, i_3 - n)$ $n = 1, 2$ (here we assume that i_3 is in the normal direction to the boundary). Thus for each point on the boundary (i_1, i_2, i_3) the values of $\mathbf{t}_\mu \cdot \mathbf{U}$ can be determined at the fictitious points $(i_1, i_2, i_3 - 1)$ and $(i_1, i_2, i_3 - 2)$. There is no coupling between adjacent

boundary points so no large system of equations need to be solved. The tangential components of the velocity are determined for all fictitious points on the entire boundary. The second step is to determine the normal component of the velocity at the fictitious points from

$$\left.\begin{aligned} \mathbf{U}_i(t)-\mathbf{u}_B(\mathbf{x}_i,t)&=0\\ \nabla_4\cdot\mathbf{U}_i(t)&=0\\ D_{4n}(\nabla_4\cdot\mathbf{U}_i(t))&=0\end{aligned}\right\} \quad \text{for} \quad i\in\text{boundary}.$$

If the grid is orthogonal to the boundary then the divergence on the boundary can be written in the form

$$\nabla\cdot\mathbf{u}=\frac{1}{e_1e_2e_3}\left\{\frac{\partial}{\partial n}(e_2e_3\mathbf{n}\cdot\mathbf{u})+\frac{\partial}{\partial t_1}(e_1e_3\mathbf{t}_1\cdot\mathbf{u})+\frac{\partial}{\partial t_2}(e_1e_2\mathbf{t}_2\cdot\mathbf{u})\right\},$$

where the e_m are functions of $\partial\mathbf{x}/\partial\mathbf{r}$. Note that only normal derivatives of $\mathbf{n}\cdot\mathbf{u}$ appear in the expression for the divergence. Thus, at a boundary point, (i_1,i_2,i_3), the stencil for $\nabla_4\cdot\mathbf{U}$ will only involve the fictitious points at (i_1,i_2,i_3-n), $n=1,2$. Similarly, the stencil for $D_{4n}(\nabla_4\cdot\mathbf{U})$ at a boundary will only involve the fictitious points at (i_1,i_2,i_3-n), $n=1,2$. Thus there is no coupling between adjacent boundary points and the unknown values for $\mathbf{n}\cdot\mathbf{u}$ can be easily determined. Note that the equations for $D_{4n}(\nabla_4\cdot\mathbf{U})$ will couple values for $\mathbf{t}_\mu\cdot\mathbf{U}$ at fictitious points along the boundary but these values have already been determined in the first step.

In practice the boundary conditions are solved in a correction mode—some initial guess is assumed for the values at the fictitious points and a correction is computed. If the grid is orthogonal or nearly orthogonal to the boundary then the first correction will give an accurate answer to the boundary conditions. If the grid is not orthogonal to the boundary then the solution procedure can repeated one or more times until a desired accuracy is achieved. This iteration should converge quickly provided that the grid is not overly skewed.

4.4. *Discrete Divergence*

The discrete divergence $\delta_i=\nabla_4\cdot\mathbf{U}_i$ will not be identically zero for the scheme described in this paper. By applying the operator $\nabla_4\cdot$ to the momentum equations it follows that δ_i will satisfy

$$\begin{aligned} \frac{d}{dt}\delta_i+(\mathbf{U}_i\cdot\nabla_4)\,\delta_i-\nu\,\Delta_4\delta_i&=\mathscr{D}_i \quad \text{for } i\in\text{the interior}\\ \delta_i&=0 \quad \text{for } i\in\text{on the boundary}\\ D_{4n}\delta_i&=0 \quad \text{for } i\in\text{on the boundary},\end{aligned}$$

where

$$\mathscr{D}_i=\left\{\nabla_4\cdot[(\mathbf{U}_i\cdot\nabla_4)\,\mathbf{U}_i]-(\mathbf{U}_i\cdot\nabla_4)\,\delta_i-\sum_{m=1}^{n_d}\nabla_4U_{m,i}\cdot D_{4x_m}\mathbf{U}_i\right\}+\{\Delta_4P_i-\nabla_4\cdot(\nabla_4P_i)\}.$$

The discrete divergence will be nonzero due to the presence of the forcing term $\mathscr{D}_i$, which will be $\mathscr{O}(h^4)$ when the solution is smoothly represented on the grid. The interpolation conditions can also act as source terms for the discrete divergence.

In order to more effectively damp out any nonzero divergence, a term proportional to δ_i can be added to the pressure equation:

$$\Delta_4P+\sum_{m=1}^{n_d}\nabla_4U_{m,i}\cdot D_{4x_m}\mathbf{U}_i-\nabla_4\cdot\mathbf{f}-C_d\nu\mathscr{V}_i\delta_i=0,$$

$$\mathscr{V}_i=\frac{1}{\Delta x_{1,i}^2}+\frac{1}{\Delta x_{2,i}^2}+\frac{1}{\Delta x_{3,i}^2}.$$

This introduces a linear damping term in the equation for the divergence

$$\frac{d}{dt}\delta_i+(\mathbf{U}_i\cdot\nabla_4)\,\delta_i-\nu\,\Delta_4\delta_i=\mathscr{D}_i-C_d\nu\mathscr{V}_i\delta_i.$$

This technique of adding a damping term is well known and has been used previously by a number of researchers in the fields of incompressible flows (the MAC method of Harlow and Welch [12]) and electromagnetics (Marder [22]). In the projection method and the MAC method, for example, a term proportional to $\delta_i/\Delta t$ is added, chosen so that the velocity field at the new time step is exactly divergence free. Although in the present situation it is not possible to make the discrete divergence exactly zero, adding such a term does reduce the divergence.

4.5. *Compatibility Conditions and Projecting the Initial Conditions*

The equation $\nabla\cdot\mathbf{u}=0$ imposes some compatibility constraints on the initial and boundary conditions. For example, the initial conditions should satisfy $\nabla\cdot\mathbf{u}_0=0$. Imposing the divergence free condition up to the boundary implies that the normal component of $\mathbf{u}_0$ should equal the normal component of the velocity specified on the boundary, $\mathbf{n}\cdot\mathbf{u}_0=\mathbf{n}\cdot\mathbf{u}_{\partial\Omega}$, see [10]. A further constraint follows by integrating $\nabla\cdot\mathbf{u}$ over Ω and using the Gauss divergence theorem, giving $\int_{\partial\Omega}\mathbf{n}\cdot\mathbf{u}\,ds=0$.

In many cases it may not be easy to generate initial conditions that satisfy the compatibility conditions. It is, however, well known how to take a given function $\mathbf{u}_I$ and to

project this function so that it is divergence-free. This projection is defined by

$$\begin{aligned} \mathbf{u}_0 &= \mathbf{u}_I + \nabla\phi, & \mathbf{x} &\in \Omega, \\ \mathbf{u}_0 &= \mathbf{u}_I, & \mathbf{x} &\in \partial\Omega \\ \Delta\phi &= -\nabla\cdot\mathbf{u}_I, & \mathbf{x} &\in \Omega \\ \mathbf{n}\cdot\nabla\phi &= 0, & \mathbf{x} &\in \partial\Omega. \end{aligned}$$

The function $\mathbf{u}_0$ will satisfy $\nabla\cdot\mathbf{u}_0 = 0$ and $\mathbf{n}\cdot\mathbf{u}_0$ will equal $\mathbf{n}\cdot\mathbf{u}_I$ on the boundary. Note, however, that this projection does not force the tangential components of $\mathbf{u}_0$ to be continuous at the boundary.

In the discrete case this projection is easy to compute since the function ϕ satisfies the same equation as the pressure, but with different data. However, the discrete projection operator does not make the discrete divergence exactly zero for the same reason that the discrete divergence is not exactly zero in the overall scheme. Thus if the initial function, $\mathbf{u}_I$, is not smooth then it may be necessary to first smooth $\mathbf{u}_I$ before applying the projection. In practice a sequence of smoothing and projection steps is applied until the discrete divergence no longer decreases significantly.

5. NUMERICAL RESULTS

The method described in this paper has been implemented in a Fortran program called CGINS (standing for Composite Grid Incompressible Navier–Stokes solver) [15]. A single computer program handles both the two-dimensional and the three-dimensional cases. The elliptic pressure equation is solved using CGES (Composite Grid Equation Solver) which is a general purpose routine for the solution of PDE boundary value problems [13] or by CGMG [16], a multigrid solver for overlapping grids. Data structures and memory allocation are managed by the DSK data structure package [7]. The DSK package enables one to write codes for a general class of overlapping grids such as those created by the grid construction program CMPGRD [5, 8]. Those readers interested in obtaining a copy of these programs and the overlapping grid construction program CMPGRD, should contact the author.

TABLE I

Errors for Flow in a Square at $t = 1.0$ and Estimated Convergence Rate, $e \propto h^{\sigma}$ ($f = 1$, $\nu = 0.05$)

Grid	Error in $\mathbf{u}$	Error in p	Maximum in $\nabla\cdot\mathbf{u}$
20×20	1.2×10^{-3}	4.1×10^{-3}	1.2×10^{-2}
30×30	2.5×10^{-4}	6.8×10^{-4}	1.2×10^{-3}
40×40	7.9×10^{-5}	2.1×10^{-4}	2.4×10^{-4}
σ	3.9	4.0	5.6

TABLE II

Errors for Flow in a Rotated Square at $t = 1.0$ and Estimated Convergence Rate, $e \propto h^{\sigma}$ ($f = 1$, $\nu = 0.05$)

Grid	Error in $\mathbf{u}$	Error in p	Maximum in $\nabla\cdot\mathbf{u}$
20×20	4.2×10^{-3}	7.1×10^{-3}	2.1×10^{-2}
30×30	7.4×10^{-4}	2.1×10^{-3}	1.4×10^{-3}
40×40	2.3×10^{-4}	6.3×10^{-4}	3.8×10^{-4}
σ	4.2	3.5	5.8

Results of some numerical computations in two and three space dimensions are now presented. The primary interest of these studies is to show the fourth-order convergence of the spatial discretization. The time step is taken sufficiently small so that the errors due to the time stepping are negligible.

5.1. *Results in Two Space Dimensions*

Twilight-zone flow. Writing and debugging a large code can be difficult. It is helpful to have some exact solutions so that errors can be measured. In order to generate *exact* solutions a very useful technique is to force the equations so that the solution can be made equal to any given function. True solutions created in this way will be called *twilight-zone flows* after Brown [4]. For the initial stages of debugging it is very useful to have a true solution for which the fourth-order method should be exact. Thus for a rectangular grid a true solution which is a quadratic polynomial should be computed exactly and any errors in the program can be quickly traced. For the convergence studies presented here, however, a slightly more complicated true solution is used. In two dimensions the following twilight-zone flow is chosen:

$$\begin{aligned} \mathbf{u}_{\text{true}}(x, y, t) &= (\sin^2(fx)\sin(2fy)\cos(2\pi t), \\ &\quad -\sin(2fx)\sin^2(fy)\cos(2\pi t)) \\ p_{\text{true}}(x, y, t) &= \sin(fx)\sin(fy)\cos(2\pi t) \end{aligned}$$

In each case a value of $\nu = 0.05$ is chosen. A damping term for the divergence is added to the pressure equation, as described in Section 4.4. The coefficient of the damping is usually taken as $C_d = 1$.

TABLE III

Errors for Flow in a Circle at $t = 1.0$ and Estimated Convergence Rate, $e \propto h^{\sigma}$ ($f = \frac{1}{2}$, $\nu = 0.05$)

Grid	Error in $\mathbf{u}$	Error in p	Maximum in $\nabla\cdot\mathbf{u}$
$35\times 35 \cup 55\times 11$	3.1×10^{-3}	1.0×10^{-2}	1.7×10^{-2}
$69\times 69 \cup 109\times 21$	1.9×10^{-4}	6.4×10^{-4}	8.8×10^{-4}
σ	4.0	4.0	4.3

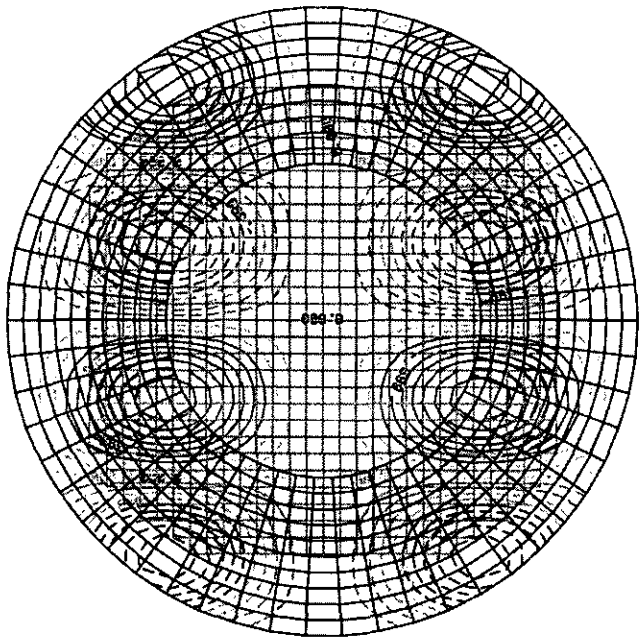

FIG. 1. Twilight-zone flow in a circle, overlapping grid, and contours of u_1.

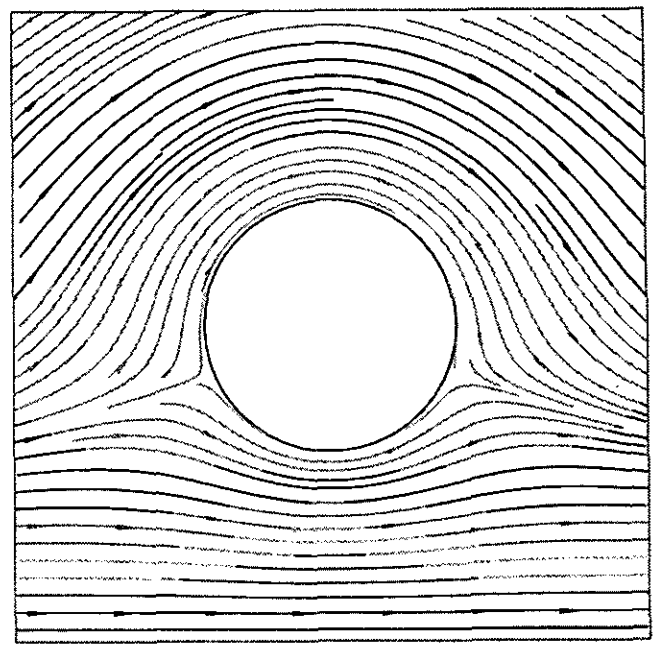

FIG. 2. Stokes flow past a cylinder near a moving wall, streamlines.

In Table I errors at $t = 1.0$ are given for solving the equations on a square with sides of length 1. Indicated are the maximum errors in $\mathbf{u}$, p, and $\nabla \cdot \mathbf{u}$. The divergence is calculated as $\nabla_4 \cdot \mathbf{U}_i$ at all interior and boundary points, although by construction (and in fact) this approximation to the divergence is zero on the boundary. The estimated convergece rate σ, error $\propto h^\sigma$, is also shown. σ is estimated by a least squares fit to the maximum errors given in the table.

In Table II the errors are given for solving the problem on a unit square which has been rotated about the origin by 45° in the clockwise direction. This is a good test as the boundary conditions are expressed in terms of the normal and tangential components of the equations. The normal and tangential velocities on the boundaries of this regions are mixtures of u_1 and u_2. The errors are different from the previous example because the true solution has not been rotated.

In Table III the errors are given for solving the equations on a region bounded by a circle of radius 1. For this example a value of $f = \frac{1}{2}$ is taken in $\mathbf{u}_{\text{true}}$ which permits a more meaningful comparison to the previous tests as the region is twice as wide. The composite grid for this domain and contour lines of the computed solution u_1 are shown in Fig. 1. For fourth-order discretizations two lines of interpolation points are used. The interpolation stencil is taken as a 5×5 square and interpolation is done in the (r_1, r_2) coordinates as described earlier in this paper.

TABLE IV

Stokes Flow, Steady Flow Past a Cylinder near a Wall

Grid	Error in $\mathbf{u}$	Maximum in $\nabla \cdot \mathbf{u}$
$(27^2) \cup (9 \times 43)$	7.3×10^{-4}	8.5×10^{-3}
$(53^2) \cup (17 \times 85)$	5.6×10^{-5}	3.8×10^{-4}
σ	3.7	4.5

5.1.1. *Comparison to an exact solution to Stokes equations.* The Stokes equations result when the nonlinear terms in the incompressible Navier–Stokes equations are set to zero. There is a nontrivial exact solution to the Stokes equations for flow between two rotating non-cocentric cylinders reported in Wannier [27]. This solution is used for code validation in [19]. For flow past a cylinder next to a moving wall (a limit as the radius of the outer cylinder tends to infinity) this exact solution is

$$u_1 = -2\frac{(A+Fy)}{k_1}\left((S+y)+\frac{k_1}{k_2}(S-y)\right) - F\log\left(\frac{k_1}{k_2}\right) - \frac{B}{k_1}\left((S+2y)-\frac{2y(S+y)^2}{k_1}\right) - \frac{C}{k_2}\left((S-2y)+\frac{2y(S-y)^2}{k_2}\right) - D$$

$$u_2 = \frac{2x}{k_1 k_2}(A+Fy)(k_2-k_1) - 2Bxy\frac{(S+y)}{k_1^2} - 2Cxy\frac{(S-y)}{k_2^2}.$$

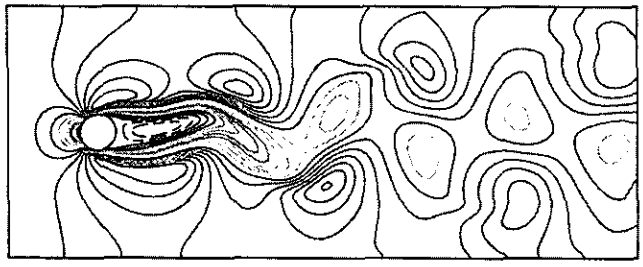

FIG. 3. Flow past a cylinder, horizontal velocity.

Here the wall, located at $y=0$, moves with speed U, the cylinder has radius R and is a distance D from the wall, and

$$S=\sqrt{D^2-R^2}, \quad k_1=x^2+(s+y)^2, \quad k_2=x^2+(s-y)^2$$

$$G=\frac{D+S}{D-S}, \quad A=-U\frac{D}{\log(G)}, \quad B=2\frac{(D+S)U}{\log(G)}$$

$$C=2\frac{(D-S)U}{\log(G)}, \quad D=-U, \quad F=\frac{U}{\log(G)}.$$

In Table IV we give the errors on two different overlapping grids for the solution to this steady state problem. In Fig. 2 we show the solution (streamlines) on the coarser grid.

5.1.2. *Unsteady flow past a cylinder.* As a final example in two space dimensions, results are shown from the computation of the flow past a cylinder. The cylinder is located at the origin and has radius $\frac{1}{2}$. The computational domain is $[x_a, x_b]\times[y_a, y_b]=[-2.5, 15]\times[-3.5, 3.5]$. The kinematic viscosity is $\frac{1}{100}$. The initial conditions are a uniform flow of $(u_1, u_2)=(1, 0)$ that are projected according to the algorithm described in Section 4.5. The top and bottom boundaries are slip walls ($\mathbf{n}\cdot\mathbf{u}=0$, $\partial_n(\mathbf{t}\cdot\mathbf{u})=0$). Here $\mathbf{n}$ is the inward facing unit normal vector and $\mathbf{t}$ is the unit tangent vector. The left boundary is inflow ($\mathbf{n}\cdot\mathbf{u}=1$, $\mathbf{t}\cdot\mathbf{u}=0$), and the right boundary outflow ($\partial_n p=-2\nu/(\frac{1}{2}(y_b-y_a))^2$). The cylinder has no-slip boundary conditions ($\mathbf{u}=0$). The Reynolds number based on the cylinder diameter and velocity 1 is $\mathrm{Re}=100$. At this Reynolds number the steady symmetric solution is unstable and an unsteady flow develops. The unsteady flow takes a long time to develop—it is not until time $t=40$ that the Kármán vortex street is clearly visible. See Figs. 3–5. The ratio of the maximum divergence to the maximum vorticity was always less than about 3×10^{-3}. In Table V we give some sample timings for this run.

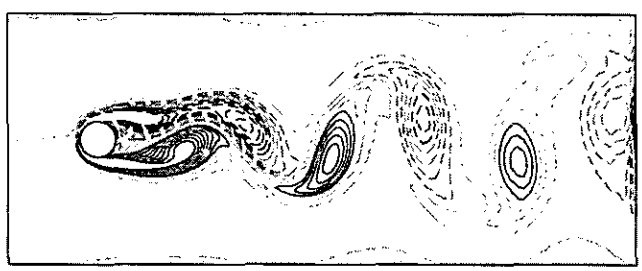

FIG. 4. Flow past a cylinder, vorticity, max = 26, min = −26.

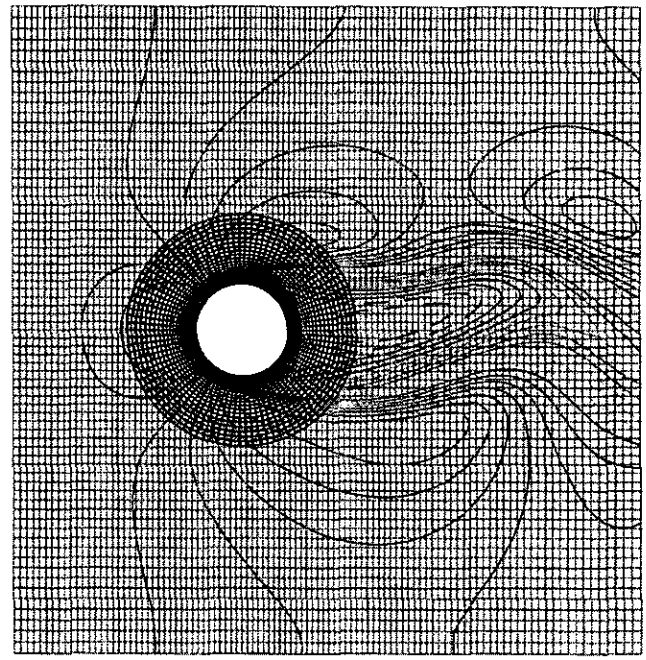

FIG. 5. Overlapping grid near the cylinder with contours of u_1.

5.2. *Results in Three Space Dimensions*

Twilight-zone flow. In three space dimensions the equations are forced so that the true solution is known and equal to

$$\mathbf{u}_{\text{true}}(x, y, z, t)=(\sin(fx)\cos(fy)\cos(fz)\cos(2\pi t),$$
$$\cos(fx)\sin(fy)\cos(fz)\cos(2\pi t),$$
$$-2\cos(fx)\cos(fy)\sin(fz)\cos(2\pi t)),$$
$$p_{\text{true}}(x, y, z, t)=\sin(fx)\sin(fy)\sin(fz)\cos(2\pi t).$$

TABLE V

Times in Seconds for Flow Past a Cylinder (IBM RS/6000-530)

	Time/step	Percentage
Pressure solve (Yale)	0.40	27
Interpolation (explicit)	0.024	2
Second-order Adams PC	0.66	43
Boundary conditions	0.11	7
Other	0.32	21
Total	1.5	100

Total number of grid points = 14,581
Fourth-order

TABLE VI

Errors for Flow in a Cube at $t = 1.0$ and Estimated Convergence Rate, $e \propto h^{\sigma}$ ($f = 1$, $\nu = 0.05$)

Grid	Error in $\mathbf{u}$	Error in p	Maximum in $\nabla \cdot \mathbf{u}$
20^3	9.6×10^{-5}	7.6×10^{-4}	8.9×10^{-4}
30^3	1.7×10^{-5}	1.3×10^{-4}	9.8×10^{-5}
σ	4.3	4.4	5.4

TABLE VII

Errors for Flow in a Spherical Shell at $t = 0.5$ and Estimated Convergence Rate, $e \propto h^{\sigma}$ ($f = \frac{1}{4}$, $\nu = 0.05$)

Grid	Error in $\mathbf{u}$	Error in p	Maximum in $\nabla \cdot \mathbf{u}$
$25^2 \times 7 \cup 25^2 \times 7$	1.1×10^{-3}	3.8×10^{-3}	2.6×10^{-3}
$37^2 \times 10 \cup 37^2 \times 10$	2.2×10^{-4}	7.0×10^{-4}	4.8×10^{-4}
$49^2 \times 13 \cup 49^2 \times 13$	7.3×10^{-5}	2.2×10^{-4}	1.7×10^{-4}
σ	4.0	4.2	4.2

In Table VI results are shown for flow in the unit cube with Dirichlet boundary conditions on all walls.

In Table VII results are shown for computations on a spherical shell. The spherical shell is the domain outside a sphere of radius $R_0 = \frac{1}{2}$ and inside a concentric sphere of radius $R_1 = 1$. The grid for this region was created using two component grids; one component grid covers the top half of the domain and the second grid covers the bottom half.

Flow past a sphere. As a final example we show the flow past a sphere in a channel. The sphere has radius 1 with centre at (0, 0, 0). The channel is the domain $[x_a, x_b] \times [y_a, y_b] \times [z_a, z_b] = [-2.2, 2.2]^3$. The overlapping grid consists of three component grids; two patches cover the sphere and one rectangular grid fills the channel. The number of grid points on each grid is $(35^3) \cup (25^2 \times 9) \cup (25^2 \times 9)$ and there are a total of 47,550 active grid points. The initial conditions are a uniform flow of $\mathbf{u} = (1, 0, 0)$ that are projected according to the algorithm described in Section 4.5. The velocity is specified as inflow on the $x = x_a$ face of the channel ($\mathbf{u} = (1, 0, 0)$); the side walls have slip boundary conditions ($\mathbf{n} \cdot \mathbf{u} = 0$, $\partial_n(\mathbf{t}_\mu \cdot \mathbf{u}) = 0$); and the face

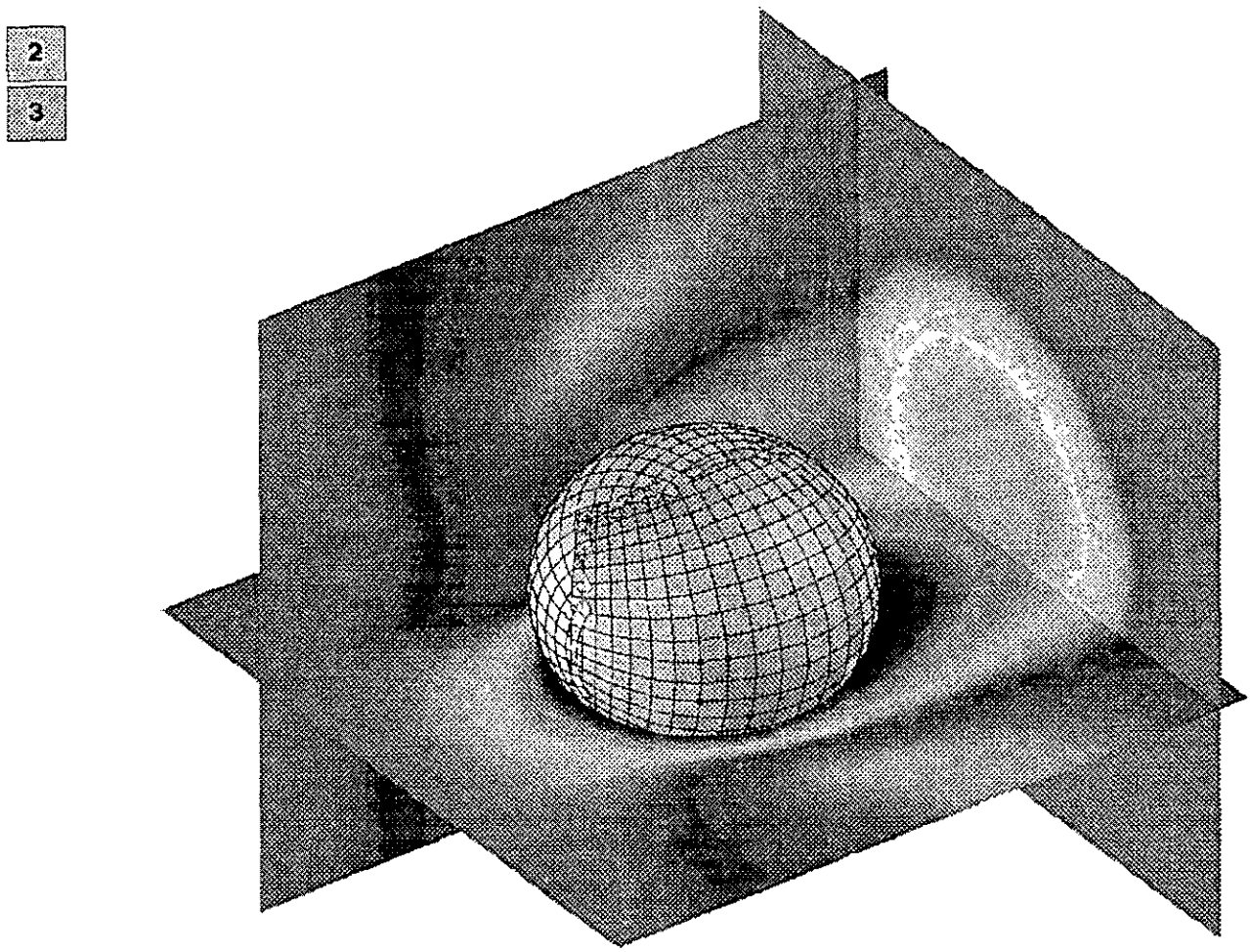

FIG. 6. Flow past a sphere, contours of u_1 at $t = 2$ ($\nu = 0.05$).

TABLE VIII

Times in Seconds for Flow Past a Sphere (IBM RS/6000-530)

	Time/step	Percentage
Pressure solve (GMRES)	19.0	42
Interpolation (implicit)	7.7	17
Second-order Adams PC	13.0	28
Boundary conditions	4.1	9
Other	1.2	3
Total	45.0	100
Total number of grid points = 17,550		
Fourth-order		

$x = x_b$ is outflow ($\frac{1}{10}\partial_n p + p = \text{given}$). The surface of the sphere has no-slip boundary conditions ($\mathbf{u} = 0$). The pressure equation was solved with GMRES. Figure 6 shows contours of u_1 on some planes that pass through the computational domain. Table VIII gives timings for this run. Since the grid is quite coarse, the number of interpolation points (10,728) is a high percentage of the total number of points. This explains why the interpolation takes so much time.

ACKNOWLEDGMENTS

The author thanks Professor Heinz-Otto Kreiss for many valuable discussions, Dr. Geoffrey Chesshire for help with overlapping grids, and Dr. Michael Henderson for help with the graphics. This work was partially supported by Grant N00014-91-C-0081 from the Office of Naval Research.

REFERENCES

1. S. Abdallah, *J. Comput. Phys.* **70**, 182 (1987).
2. T. J. Baker, "Generation of Tetrahedral Meshes around Complete Aircraft," in *Numerical Grid Generation in Computational Fluid Mechanics'88*, edited by S. Sengupta (Pineridge, Swansea, UK, 1989), p. 675.
3. J. B. Bell, P. Colella, and H. M. Glaz, *J. Comput. Phys.* **85**, 257 (1989).
4. D. L. Brown, private communication, 1987.
5. D. L. Brown, G. Chesshire, and W. D. Henshaw, Report LA-UR-89-1294, Los Alamos National Laboratory, 1989 (unpublished).
6. G. Browning, *Mon. Weather Rev.* **117**, 1058 (1989).
7. G. Chesshire and W. D. Henshaw, IBM Research Report RC 14353 (IBM Research Division, Yorktown Heights, NY, 1988).
8. G. Chesshire and W. D. Henshaw, *J. Comput. Phys.* **90**, 1 (1990).
9. A. J. Chorin, *Math. Comput.* **22**, 745 (1968).
10. P. M. Gresho, *Adv. Appl. Mech.* **28** (1992).
11. P. M. Gresho and R. L. Sani, *Int. J. Numer. Methods Fluids* **7**, 1111 (1987).
12. F. Harlow and J. Welch, *J. Comput. Phys.* **8**, 2182 (1965).
13. W. D. Henshaw IBM RC 19361, IBM Research Division, Yorktown Heights, NY, 1994.
14. W. D. Henshaw, IBM RC 19360, IBM Research Division, Yorktown Heights, NY, 1994.
15. W. D. Henshaw, IBM RC 19359, IBM Research Division, Yorktown Heights, NY, 1994.
16. W. D. Henshaw and G. Chesshire, *SIAM J. Sci. Stat. Comput.* **8**, 914 (1987).
17. W. D. Henshaw, G. Chesshire, and M. Henderson, IBM research report, IBM Research Division, Yorktown Heights, NY, 1992 (unpublished).
18. W. D. Henshaw, H. Kreiss, and L. Reyna, *Comput. Fluids* **23**, 575 (1994).
19. G. Karniadakis, M. Israeli, and S. A. Orszag, *J. Comput. Phys.* **97**, 414 (1991).
20. J. Kim and P. Moin, *J. Comput. Phys.* **59**, 308 (1985).
21. C. Kiris, S. Rogers, D. Kwak, and I. Chang, *ASME J. Biofluidmech. Eng.*, to appear.
22. B. Marder, *J. Comput. Phys.* **68**, 48 (1987).
23. J. L. Steger and J. A. Benek, *Comput. Methods Appl. Mech. Eng.* **64**, 301 (1987).
24. J. Strikwerda, *SIAM J. Sci. Stat. Comput.* **5**, 56 (1984).
25. J. F. Thompson, *AIAA J.* **26**, 271 (1988).
26. J. Y. Tu and L. Fuchs, *Int. J. Numer. Methods Fluids* **15**, 693 (1992).
27. G. H. Wannier, *Quart. Appl. Math.* **8**, 1 (1950).
28. M. J. Ward, W. D. Henshaw, and J. B. Keller, *SIAM J. Appl. Math.* **53**, 799 (1993).
29. J. A. Wright and W. Shyy, *J. Comput. Phys.* **107**, 225 (1993).