JOURNAL OF COMPUTATIONAL PHYSICS 116, 79–96 (1995)

Domain Decomposition and the Compact Fourth-Order Algorithm for the Navier–Stokes Equations

J. ROKICKI* AND J. M. FLORYAN

Department of Mechanical Engineering, The University of Western Ontario, London, Ontario, Canada N6A 5B9

Received March 8, 1993

We consider a fourth-order, compact finite-difference method for the Navier–Stokes equations using the streamfunction–vorticity formulation. Various algebraic boundary formulas for vorticity are investigated including new implicit formulas of the third and fourth order. An algorithm for determination of pressure from a suitable Poisson equation is given. Results of various tests show that the error of the algorithm is proportional to $Re^2 \cdot h^4$. Domain decomposition coupled with multiprocessing was investigated as a method for acceleration of computations. It is shown that the acceleration approaches the theoretical maximum.

1. INTRODUCTION

The main objective of the present work is to develop a fast and accurate algorithm that is capable of accurately predicting medium to large Reynolds number flows in non-standard geometries. Second-order discretization schemes (e.g., classic [1]) require so many grid points for accurate calculations that their application to realistic flow problems becomes impractical. Spectral methods provide high accuracy but have difficulties in handling non-standard geometries. The present work focuses on higher-order compact finite-difference methods which are sufficiently flexible to deal with complicated geometries and can provide the required accuracy at an acceptable computational cost. The streamfunction–vorticity formulation is selected to bypass direct evaluation of pressure.

A number of implementations of compact differencing to the streamfunction–vorticity formulation of the flow problem can be found in the existing literature. A spline collocation procedure is used in [2], higher-order equations are split into a system of equations of first order in [3], and first derivatives are completely eliminated as independent variables in [4–6]. Different procedures to obtain boundary formulas for vorticity range from simple algebraic formulas [4, 7, 3] to more complex ones based on non-local orthogonal projection techniques [8–10].

* Permanent address: Institute of Aeronautics and Applied Mechanics, Warsaw University of Technology, Nowowiejska 22/24, 00-665 Warsaw, Poland.

The algorithm presented here is based on the method of Dennis and Hudson [4] with correction for the vorticity boundary formula. The issue of an accurate treatment of boundary conditions is discussed in detail because of its importance for the domain decomposition method. The algorithm is extended to include fourth-order accurate determination of pressure from the known vorticity and velocity fields.

Verification of highly accurate algorithms is difficult because their performance is strongly affected by even small inconsistencies in the test problems. Acceptable tests are hard to identify due to the lack of non-trivial exact solutions of the Navier–Stokes equations. The most popular tests involving flow in a cavity are inappropriate because of singularities in the cavity's corners which mask the true behaviour of the algorithm. Because of the lack of sufficient testing, the existing fourth-order algorithms can be considered as being only qualitatively verified (see Section 2.1.E) and, therefore, may not provide the accuracy as claimed.

Two types of tests are used in the present work. In the first one, numerical solutions are compared directly with exact, specially constructed, "artificial" solutions of the Navier–Stokes equations. In the second one, the performance of the algorithm is verified by repeating calculations on a sequence of grids and estimating the true error and its variations as a function of grid step size h. While the exact solution does not need to be known in the latter case, its success hinges on the problem specification, which has to guarantee a sufficient degree of smoothness of the solution.

Typical, realistic problems in fluid dynamics require enormous amounts of computer work (as measured by the execution time). With the fixed processor speed, significant acceleration can be obtained only by using several processors working in parallel. The full advantage of this approach can be achieved only if the numerical algorithm itself is suitable for parallelization. The domain decomposition method is well suited for parallelization of a very broad class of problems. The method consists of dividing the computational domain into overlapping subdomains and solving the original problem on each subdomain separately, with the appropriate transfer of boundary information between the neighbouring subdomains, with each sub-

0021-9991/95 $6.00

domain being served by a different processor. This approach was used [11–14] in the case of primitive variables formulation and second-order accurate methods but, to the best of our knowledge, has not been extended either to the streamfunction–vorticity formulation or to the fourth-order methods. Gajjar [15] investigated parallelism in the case of streamfunction–vorticity formulation, but without using domain decomposition.

The algorithm described in this paper takes advantage of the domain decomposition principles and thus is suitable for implementation on several processors working concurrently. Various practical and theoretical problems arising from multiprocessor implementation are analysed for a model problem. The acceleration of computations is tested in numerical experiments with the multiprocessor-computer being simulated using a conventional one.

The paper is organized as follows. Section 2 describes single domain (single processor) implementation of the algorithm. Section 2.1 gives discretization formulas and their numerical verification. Section 2.2 describes the solution to the pressure problem. Section 3 deals with the multidomain (multiprocessor) implementation. Section 4 gives a short summary of the main conclusions.

2. SINGLE DOMAIN (SINGLE PROCESSOR) IMPLEMENTATION

2.1. Flow Problem

2.1.A. *Preliminaries*

The Navier–Stokes equations written in the streamfunction–vorticity formulation for a two-dimensional incompressible flow in a cartesian reference system (x, y) are

$$\Delta\zeta = \mathrm{Re}\,(u\zeta_x + v\zeta_y - \mathrm{curl}\,\mathbf{f}_e), \tag{2.1a}$$

$$\Delta\psi = -\zeta, \tag{2.1b}$$

$$u = \psi_y, \quad v = -\psi_x, \tag{2.1c}$$

where $\zeta = v_x - u_y$ is the vorticity, ψ is the streamfunction, (u, v) denote velocity components in the (x, y) direction, $\mathbf{f}_e = (f_{e1}, f_{e2})$ is the known external body force and $\mathrm{curl}\,\mathbf{f}_e = \partial f_{e2}/\partial x - \partial f_{e1}/\partial y$. The usual, natural conditions for the streamfunction ψ and its normal derivative $\partial\psi/\partial\mathbf{n}$ on the boundary Γ are

$$\psi|_\Gamma = a(t), \quad t \in \Gamma \tag{2.2a}$$

$$\partial\psi/\partial\mathbf{n}|_\Gamma = b(t). \tag{2.2b}$$

It is assumed that all the necessary tangential derivatives of $a(t)$ and $b(t)$ exist and are known. We also consider the auxiliary boundary conditions of the form

$$\psi|_\Gamma = a(t), \quad t \in \Gamma \tag{2.3a}$$

$$\zeta|_\Gamma = c(t), \tag{2.3b}$$

where $a(t)$ is the same as in (2.2a) while $c(t)$ can be arbitrary. There are cases where boundary conditions (2.3), i.e., prescribing vorticity at the boundary, are perfectly natural as, for example, on the line of symmetry of a flow or for periodic flows. These boundary conditions appear to be natural also in the case of domain decomposition (Section 3).

Problem (2.1) with boundary conditions (2.3) can be solved directly. Problem (2.1) with boundary conditions (2.2) can be solved using boundary conditions (2.3) and determining iteratively $c(t)$, so that the boundary condition (2.2b) is also satisfied.

The solution algorithm considered here has a classical structure. Its main steps are listed below in order to simplify discussion in the rest of the paper. These steps are (regardless of the discretization type):

1. Initialize vorticity field ζ in the flow region Ω and at the boundary Γ.

2. Solve Poisson equation (2.1b) for the streamfunction with the Dirichlet boundary conditions (2.2a), i.e.,

$$\Delta\psi = -\zeta, \quad \psi|_\Gamma = a(t), \quad t \in \Gamma. \tag{2.4}$$

3. Calculate velocity from streamfunction using (2.1c).

4. Correct the boundary value ζ_B of vorticity in order to satisfy the Neumann boundary condition (2.2b).

5. Solve the linear PDE for vorticity (2.1a) with the Dirichlet boundary conditions determined in step 4:

$$\Delta\zeta = \mathrm{Re}(u\zeta_x + v\zeta_y - \mathrm{curl}\,\mathbf{f}_e), \quad \zeta|_\Gamma = \zeta_B. \tag{2.5}$$

6. Check for the convergence of vorticity ζ and streamfunction ψ. If there is no convergence proceed to step 2.

In the following section, a fourth-order finite-difference discretization scheme for systems (2.4) and (2.5), together with a suitable formula for calculation of velocity from (2.1c), are presented. The various boundary correction formulas for vorticity are investigated in Section 2.1.C.

2.1.B. *Discretization of the Field Equations*

We restrict our considerations to the flow region Ω in the form of a unit square $0 \le x \le 1$, $0 \le y \le 1$. A uniform grid of constant step size h in the x and y directions is chosen. Locations of the grid points are defined as $[(i-1)h, (j-1)h]$, $i, j = 1, \ldots, N+1$, $h = 1/N$.

Consider a general, linear convection–diffusion equation in the form:

$$\phi_{xx} + \phi_{yy} - (\tilde{u}\phi_x + \tilde{v}\phi_y) = R, \tag{2.6}$$

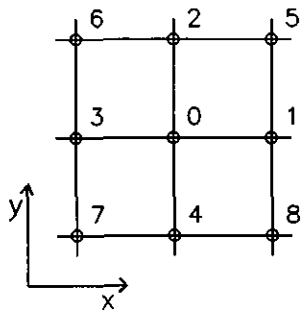

FIG. 1. Sketch of a typical computational molecule in the interior of the solution domain.

where the lower index denotes a differentiation operator, and the functions $\tilde{u}$, $\tilde{v}$, and R are prescribed. Following Dennis and Hudson [4], we replace (2.6) with its h^4-accurate, nine point, finite-difference approximation to get

$$d_1\phi_1 + \cdots + d_8\phi_8 - d_0\,\phi_0 + B_0 = 0, \tag{2.7}$$

where the Southwell index notation is used as indicated in Fig. 1. A similar scheme was reported earlier by Gupta [5, 6]. The coefficients d_n and B_0 in (2.7) are given by the formulas:

$$\begin{aligned}
d_0 &= 40 + 2h^2(\tilde{u}_0^2 + \tilde{v}_0^2) - 4h^2(\tilde{u}_{x0} + \tilde{v}_{y0}) \\
d_{1/3} &= 8 \mp 4h\tilde{u}_0 + h^2(\tilde{u}_0^2 - 2\tilde{u}_{x0}) \\
&\quad \pm h^3[\tilde{u}_0\tilde{u}_{x0} + \tilde{v}_0\tilde{u}_{y0} - (\Delta\tilde{u})_0]/2 \\
d_{2/4} &= 8 \mp 4h\tilde{v}_0 + h^2(\tilde{v}_0^2 - 2\tilde{v}_{y0}) \\
&\quad \pm h^3[\tilde{u}_0\tilde{v}_{x0} + \tilde{v}_0\tilde{v}_{y0} - (\Delta\tilde{v})_0]/2 \\
d_{5/7} &= 2 \mp h(\tilde{u}_0 + \tilde{v}_0) + \tilde{u}_0\tilde{v}_0h^2/2 - H_0h^2/2 \\
d_{6/8} &= 2 \pm h(\tilde{u}_0 - \tilde{v}_0) - \tilde{u}_0\tilde{v}_0h^2/2 + H_0h^2/2 \\
B_0 &= -h^2[8R_0 + (1 - h\tilde{u}_0/2)R_1 + (1 - h\tilde{v}_0/2)R_2 \\
&\quad + (1 + h\tilde{u}_0/2)R_3 + (1 + h\tilde{v}_0/2)R_4],
\end{aligned} \tag{2.8}$$

where $\Delta = \partial^2/\partial x^2 + \partial^2/\partial y^2$, $H = \partial\tilde{v}/\partial x + \partial\tilde{u}/\partial y$. The numerical scheme (2.7)–(2.8) can be used for discretization of the vorticity transport equation (2.5), using the following substitutions:

$$\tilde{u} = \mathrm{Re}\cdot u, \quad \tilde{v} = \mathrm{Re}\cdot v, \quad \phi = \zeta, \quad R = -\mathrm{Re}\cdot \mathrm{curl}\,\mathbf{f}_e. \tag{2.9}$$

The first and second derivatives of velocity in (2.8) can be calculated with the usual second-order finite-difference formulas [4]. The same applies to the divergence of velocity field $(\tilde{u}_{x0} + \tilde{v}_{y0})$ which has to be calculated in spite of the fact that for incompressible flow it vanishes everywhere. The velocity components (u, v) have to be evaluated with fourth-order accuracy (cf. [6]), however.

Velocity evaluation constitutes a problem of calculation of a derivative of a function whose Laplacian is known. Use of Taylor expansion gives

$$\begin{aligned}
u_0 &= \psi_{y0} = (\psi_2 - \psi_4)/2h - \psi_{3y0}h^2/6 + O(h^4), \\
\psi_{3y0} &= \Delta\psi_{y0} - \psi_{2xy0} = -\zeta_{y0} - \psi_{2xy0},
\end{aligned}$$

where $\psi_{3y0} \equiv \partial^3\psi/\partial y^3$ and $\psi_{2xy0} \equiv \partial^3\psi/\partial x^2\partial y$, with both derivatives being evaluated at point 0 (see Fig. 1). Replacing ζ_{y0} and ψ_{2xy0} with their central, finite-difference, h^2-accurate approximations we obtain

$$\begin{aligned}
u_0 = \psi_{y0} &= (\psi_2 - \psi_4)/2h + (\zeta_2 - \zeta_4)h/12 \\
&\quad + (\psi_5 - 2\psi_2 + \psi_6 - \psi_8 + 2\psi_4 - \psi_7)/12h + O(h^4).
\end{aligned} \tag{2.10a}$$

Similarly,

$$\begin{aligned}
-v_0 = \psi_{x0} &= (\psi_1 - \psi_3)/2h + (\zeta_1 - \zeta_3)h/12 \\
&\quad + (\psi_5 - 2\psi_1 + \psi_8 - \psi_6 + 2\psi_3 - \psi_7)/12h + O(h^4).
\end{aligned} \tag{2.10b}$$

The system of linear equations for vorticity resulting from (2.7)–(2.9) is solved iteratively by the Gauss–Seidel relaxation procedure

$$\zeta_0^{(m+1)} = (1 - \omega)\zeta_0^{(m)} + \omega[d_1\zeta_1^{(*)} + \cdots + d_8\zeta_8^{(*)} + B_0]/d_0, \tag{2.11}$$

where ω is the relaxation factor, the upper subscript refers to the iteration number and * denotes the most recent value of ζ.

The matrix of coefficients corresponding to the system of linear equations (2.7)–(2.9) may not be, in general, diagonally dominant. This may happen when the Reynolds number is large and the step size is not small enough [4]. We found out, through numerical experiments, that the present algorithm converges even if the diagonal dominance is lost.

The Poisson equation (2.4) for the streamfunction ψ can be discretized in the same way as the equation for vorticity by using (2.7) and (2.8) and taking

$$\phi = \psi, \quad R = -\zeta, \quad \tilde{u} = \tilde{v} \equiv 0. \tag{2.12}$$

The resulting system of linear equations is solved using the same relaxation procedure with ζ replaced by ψ in (2.11). The relaxation factors for ψ and ζ are in general different.

In the actual calculations, only a few Gauss–Seidel iterations of the discretized vorticity (2.5) and streamfunction (2.4) equations are to be carried out at each step of the external iteration procedure. If Eqs. (2.4) and (2.5) were solved exactly at each step, the general algorithm would become very unstable and significant underrelaxation would be required for the vorticity boundary formula.

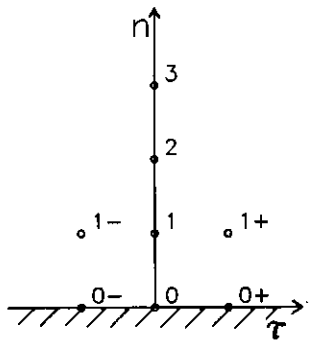

FIG. 2. Sketch of a typical computational molecule on a boundary of the solution domain.

2.1.C. *Discretization of Boundary Formula for Vorticity*

The fourth step of the algorithm (Section 2.1.A) requires updating the boundary value of vorticity in order to satisfy condition (2.2b) for the normal derivative of streamfunction. Different classical second-order formulas are discussed in [16, 7]. The fourth-order formula given by Dennis and Hudson [4] is in fact only second-order accurate, as shown later in this section. The true fourth- and third-order formulas will be derived below.

Consider a vicinity of a point on the boundary (Fig. 2). Use of Taylor expansions gives

$$(8\psi_1 - 7\psi_0 - \psi_2)/h^2 - 6\psi_{n0}/h = 2\psi_{2n0} - \psi_{4n0}h^2/3 - \psi_{5n0}h^3/5 + O(h^4), \quad (2.13)$$

where ψ_{4n0} denotes $\partial^4\psi/\partial\mathbf{n}^4$ at point "0" (Fig. 2) and other subscripts have a similar meaning. Higher normal derivatives of the streamfunction are eliminated using (2.1b), i.e.,

$$\psi_{2n0} = -\zeta_0 - \psi_{2\tau 0}, \quad \psi_{4n0} = -\zeta_{2n0} + \zeta_{2\tau 0} + \psi_{4\tau 0},$$
$$\psi_{5n0} = -\zeta_{3n0} + \zeta_{n2\tau 0} + \psi_{n4\tau 0}, \quad (2.14)$$

where $\psi_{2\tau 0}$ denotes $\partial^2\psi/\partial\tau^2$ (τ—tangential direction) at point "0" and other subscripts have a similar meaning. Substitution of (2.14) into (2.13) results in

$$[(8\psi_1 - 7\psi_0 - \psi_2)/h^2 + P] = -2\zeta_0 + \zeta_{2n0}h^2/3 + \zeta_{3n0}h^3/5 - \zeta_{2\tau 0}h^2/3 - \zeta_{n2\tau 0}h^3/5 + O(h^4), \quad (2.15)$$

where

$$P = -6\psi_{n0}/h + 2\psi_{2\tau 0} + \psi_{4\tau 0}h^2/3 + \psi_{n4\tau 0}h^3/5 \quad (2.16)$$

is known at the boundary (it vanishes if the velocity is zero there). Neglecting terms $O(h^2)$ in (2.15) results in the well-known second-order Jensen formula [16], i.e.,

$$-2\zeta_0 = (8\psi_1 - 7\psi_0 - \psi_2)/h^2 - 6\psi_{n0}/h + 2\psi_{2\tau 0}. \quad (2.17)$$

Normal derivatives ζ_{2n0} and ζ_{3n0} can be eliminated from (2.15) by expressing ζ at points 1, 2, and 3 in terms of Taylor expansions centered at point "0," i.e.,

$$3\zeta_1 - 3\zeta_2 + \zeta_3 = \zeta_0 + h^3\zeta_{3n0} + O(h^4),$$
$$5\zeta_1 - 4\zeta_2 + \zeta_3 = 2\zeta_0 - h^2\zeta_{2n0} + O(h^4).$$

This results in

$$15[(8\psi_1 - 7\psi_0 - \psi_2)/h^2 + P] + (16\zeta_1 - 11\zeta_2 + 2\zeta_3) = -23\zeta_0 - (5h^2\zeta_{2\tau 0} + 3h^3\zeta_{n2\tau 0}) + O(h^4). \quad (2.18)$$

Dennis and Hudson [4] considered a special case of zero velocity at the boundary, i.e., $\psi = u = v = 0$ ($\psi_0 = 0$, $P = 0$), and obtained boundary formula (Eq. (59) in [4]) in the form

$$15(8\psi_1 - \psi_2)/h^2 + (16\zeta_1 - 11\zeta_2 + 2\zeta_3) = -23\zeta_0. \quad (2.19)$$

The above formula agrees with (2.18) only when $-\zeta_{2\tau 0} \equiv \psi_{2n2\tau 0} = 0$ and $-\zeta_{n2\tau 0} \equiv \psi_{3n2\tau 0} = 0$, which is not the case in general. Comparison of (2.19) and (2.18) shows that the Dennis and Hudson's formula is only second-order accurate. This conclusion will be verified using numerical experiments discussed in Section 2.1.E.

The true fourth-order accurate formula is obtained by approximating $\zeta_{2\tau 0}$ with the second-order and $\zeta_{n2\tau 0}$ with the first-order finite-difference formulas and substituting them into (2.18). Considering values of vorticity ζ_0, ζ_{0+}, ζ_{0-} at all boundary points (see Fig. 2) as unknown, the final equation can be written as

$$2\zeta_{0+} + 19\zeta_0 + 2\zeta_{0-} = -15[(8\psi_1 - 7\psi_0 - \psi_2)/h^2 + P] - (10\zeta_1 - 11\zeta_2 + 2\zeta_3) - 3(\zeta_{1-} + \zeta_{1+}) + O(h^4). \quad (2.20)$$

In the case of a grid point located next to a corner (see Fig. 3), where ζ_{0+} is known but ζ_{1+} is not, Eq. (2.20) becomes

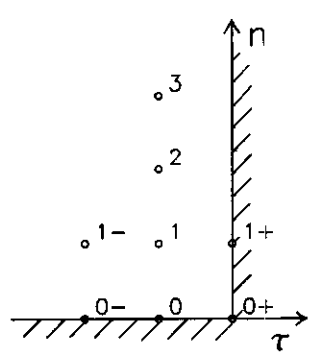

FIG. 3. Sketch of a typical computational molecule in the immediate neighbourhood of a corner of the solution domain.

$$3\zeta_{1+} + 19\zeta_0 + 2\zeta_{0-} = -15[(8\psi_1 - 7\psi_0 - \psi_2)/h^2 + P] - (10\zeta_1 - 11\zeta_2 + 2\zeta_3) - 3\zeta_{1-} - 2\zeta_{0+} + O(h^4). \quad (2.20a)$$

One may note that when the boundary conditions specified along the walls meeting at a corner are such that ζ_{0+} becomes singular, the accuracy of the approximation may deteriorate in an unpredictable manner.

Equations (2.20) written for each grid point along the boundary (except corners) result in a system of linear equations with a tridiagonal, symmetric, and diagonally dominant matrix of coefficients, which guarantees existence of its solution. The numerical cost of obtaining the solution is negligibly small in comparison with the cost of a single iteration of the discretized field equations.

Other fourth-order accurate boundary formulas for vorticity can be obtained by noting that the Laplacian of vorticity $\Delta\zeta \equiv \zeta_{2\tau} + \zeta_{2n}$ is known at the boundary from the momentum equation (2.1a). One can replace (2.14) with

$$\psi_{4n0} = -(1 + \alpha)\zeta_{2n0} + (1 - \alpha)\zeta_{2\tau 0} + \alpha(\Delta\zeta)_0 + \psi_{4\tau 0},$$
$$\psi_{5n0} = -(1 + \beta)\zeta_{3n0} + (1 - \beta)\zeta_{n2\tau 0} + \beta(\Delta\zeta)_{n0} + \psi_{n4\tau 0},$$

where α, β are arbitrary real constants. Values of vorticity at the internal points (ζ_3 or/and ζ_2) appearing in the boundary formula (2.20) can be eliminated by conscientious selection of α and β. All formulas, regardless of the particular values of α and β, are fourth-order accurate. We have determined through numerical experiments that none of them is significantly superior to the original one (2.20).

The third-order formulas can be obtained by neglecting $O(h^3)$ terms in (2.15) and (2.16). Approximation of the normal and tangential derivatives of vorticity in the form

$$\zeta_{2n0} = (\zeta_0 - 2\zeta_1 + \zeta_2)/h^2 + O(h),$$
$$\zeta_{2\tau 0} = (\zeta_{0-} - 2\zeta_0 + \zeta_{0+})/h^2 + O(h^2)$$

leads for the boundary formula

$$\zeta_{0+} + 3\zeta_0 + \zeta_{0-} = -3P - [2\zeta_1 - \zeta_2] + O(h^3), \quad (2.21)$$

whose assembly over all boundary points (without corners) results in a system of equations with a tridiagonal, symmetric, diagonally dominant coefficient matrix.

It is worth noting, that formulas with accuracy $O(h^k)$ with respect to ζ satisfy the natural boundary condition (2.2b) with accuracy $O(h^{k+1})$ (cf. (2.15)–(2.16)). Since it is the accuracy of the original problem (2.1)–(2.2) that is of interest, both the third- and the fourth-order boundary formulas can be used in order to get a fourth-order algorithm.

2.1.D. *Verification Methodology*

We employ two different methods to verify accuracy of the proposed algorithm. The best and simplest method is to compare the computed result with an exact solution. Alternatively, one may investigate properties of the computed solution as a function of grid size and to compare them with the analytical error estimates. We begin our discussion with the former one.

The known exact solutions of Navier–Stokes equations lead to a trivial form of the nonlinear terms and are, therefore, too simple to be used for meaningful verification purposes; it is the approximation of the nonlinear terms that is of crucial importance for performance of the whole algorithm. Useful tests can be carried out, however, by constructing artificial solutions, in a manner somewhat similar to [4, 17].

Let ψ be an arbitrary (and sufficiently smooth) function of two variables. If (u, v) is taken as "velocity" defined by (2.1c), ζ as "vorticity" defined by (2.1b), then introduction of the (fictitous) "body force" $\mathbf{f}_e$ as

$$\operatorname{curl} \mathbf{f}_e = -\Delta\zeta/\mathrm{Re} + u\zeta_x + v\zeta_y \quad (2.22)$$

satisfies (2.1a) identically. Although one does not need to solve (2.22) for $\mathbf{f}_e$, the solution always exists provided the right-hand side of (2.22) is sufficiently regular. This may be shown by noting that substitution $\mathbf{f}_e = [\partial\Phi_1/\partial y, -\partial\Phi_1/\partial x] + \mathbf{grad}\ \Phi_2$ reduces (2.22) to the Poisson equation for Φ_1, with Φ_2 being arbitrary.

The test examples should be selected so that (i) derivatives of ψ are easily evaluated analytically, (ii) $u\zeta_x + v\zeta_y$ does not vanish everywhere, (iii) boundary conditions are fulfilled, and (iv) the "flow" is not symmetric (to make the problem more general). As an example consider the following construction. Take

$$\psi(x, y) = f(x, y) \cdot g(x, y), \quad (2.23a)$$

where

$$f(x, y) = y \cdot a(x) + (1 - y) \cdot a(2x), \quad (2.23b)$$
$$g(x, y) = x \cdot \gamma_1 \cdot [a(y) + \delta \cdot b(y)] + (1 - x) \cdot \gamma_2 \cdot [a(2y) + \delta \cdot b(y)],$$
$$a(t) = 1 - \cos(2\pi t) + \sin(2\pi t) \cdot (1 - \cos(2\pi t))/10,$$
$$b(t) = 2 \cdot (1 - \cos(\pi t)) \cdot \sin(\pi t), \quad (2.23c)$$

and γ_1, γ_2, δ are arbitrary constants. Functions $a(x, t)$ and $b(x, t)$ have the properties

$$a(0) = a(k) = a'(0) = a'(k) = 0,$$
$$b(0) = b(2k - 1) = b'(0) = 0,$$
$$b'(2k - 1) = -4\pi,$$

for $k = 1, 2, \ldots$, and prime denotes differentiation with respect to t. As a result, ψ and $\partial\psi/\partial\mathbf{n}$ vanish on the boundary of the unit square when $\delta = 0$, while if δ is not zero, $\psi = 0$ on the boundary but the tangent velocity ($v_\tau = -\partial\psi/\partial\mathbf{n}$) does not vanish on the upper wall.

In the second method of verification, the exact solution does not need to be known, which is the case in most practical applications. The accuracy of the algorithm can be tested by repeating calculations on a sequence of grids (with diminishing grid step size) and comparing the error of the computed solution with its analytical estimates.

Let the grid sequence with steps h_k be defined as

$$h_k := h_0 q^k, \quad 0 < q < 1, \quad k = 1, 2, \ldots,$$
$$\mathbf{N}_k := 1/h_k + 1. \tag{2.24}$$

Let $\Phi(h_k)$ denote the solution obtained on the kth grid and let this solution be suitably extended onto the whole solution domain. The following propositions hold:

1. If the sequence $\Phi(h_k)$ converges to some $\Phi(0)$ and $C > 0$ exists such that

$$\forall k \geq 0, \quad \|\Phi(h_k) - \Phi(0)\| \leq C h_k^\alpha,$$
$$\alpha > 0, \quad h_k := h_0 q^k, \quad 0 < q < 1,$$

then

$$\forall k \geq 0, \quad \|\Phi(h_{k+1}) - \Phi(h_k)\| \leq (1 + q^\alpha) C h_k^\alpha$$
$$(\|\Phi(h_{k+1}) - \Phi(h_k)\| \leq \|\Phi(h_{k+1}) - \Phi(0)\|$$
$$+ \|\Phi(h_k) - \Phi(0)\| \leq (1 + q^\alpha) C h_k^\alpha),$$

where $\|\cdot\|$ denotes a suitable norm.

2. Conversely, if the sequence $\Phi(h_k) \in V$ (V-Banach space with the norm $\|\cdot\|$) has the property that

$$\exists C > 0, \forall k \geq 0, \quad \|\Phi(h_{k+1}) - \Phi(h_k)\| \leq C h_k^\alpha,$$
$$\alpha > 0, \quad h_k := h_0 q^k, \quad 0 < q < 1, \tag{2.25}$$

then the limit $\Phi(0) \in V$ exists and

$$\forall k \geq 0, \quad \|\Phi(h_k) - \Phi(0)\| \leq C h_k^\alpha/(1 - q^\alpha). \tag{2.26}$$

To prove that the limit exists observe that V is complete and that the Cauchy sequence converges geometrically, i.e.,

$$\forall k, s > 0, \quad \|\Phi(h_{k+s+1}) - \Phi(h_k)\|$$
$$\leq \|\Phi(h_{k+s+1}) - \Phi(h_{k+s})\|$$
$$+ \cdots + \|\Phi(h_{k+1}) - \Phi(h_k)\|$$
$$\leq C(h_{k+s}^\alpha + \cdots + h_k^\alpha) = C h_k^\alpha (1 + q^\alpha + \cdots + q^{s\alpha})$$
$$\leq q^{k\alpha} C h_0^\alpha/(1 - q^\alpha).$$

To prove inequality (2.26) observe that

$$\|\Phi(h_k) - \Phi(0)\| \leq \sum_{s=0}^{M} \|\Phi(h_{k+s+1}) - \Phi(h_{k+s})\|$$
$$+ \|\Phi(h_{k+M+1}) - \Phi(0)\|$$
$$\leq \sum_{s=0}^{\infty} \|\Phi(h_{k+s+1}) - \Phi(h_{k+s})\|$$
$$\leq C h_k^\alpha (1 + q^\alpha + q^{2\alpha} + \cdots)$$
$$\leq C h_k^\alpha/(1 - q^\alpha).$$

The second proposition states that it is possible to verify the order of accuracy of the discretization scheme and to estimate the error of the computed solution, without knowing the actual (exact) solution.

In the numerical experiments, we are able to check (2.25) only on a finite number of grids. The step sequences do not need to be strictly in the form (2.24). We used

$$\mathbf{N}_k = \text{NEAREST\_EVEN\_TO}[(N_0 - 1)/q^k] + 1$$
$$h_k = 1/(N_k - 1), \quad 0 < q < 1, \ k = 1, 2, \ldots, \tag{2.27}$$

with $N_0 = 11$ and $q = 2^{-1/3} \approx 0.79$. The more customary selection of $q = 2^{-1}$ would diminish the step size too rapidly to obtain the desired number of solutions. The following numerical values for N_k resulted from (2.27):

$$11, 13, 17, 21, 25, 33, 41, 51, 63, 81, 101, 125, 163, 201, 251. \tag{2.28}$$

Since we observed that the greatest error of vorticity occurred always on the boundary Γ of the solution domain, the numerical solution was extended only onto the boundary (using a fourth-order spline interpolation technique). The max and L_2 norms were approximated using a subgrid which had twice as many points as the finest grid used in the computations.

The discretization scheme will exhibit its true convergence properties only if the solution is smooth enough. This can be guaranteed by the appropriate selection of both the geometry of the solution domain and the boundary conditions, which cannot lead to the appearance of discontinuities in the solution and several of its higher derivatives. For example, it might be necessary to require that the sixth derivative of vorticity exists and is bounded—cf. error of Numerov formula [4]. The very popular computational example of a square cavity with the moving upper wall [2, 3, 5, 6, 15] has a solution with discontinuous velocity and infinite vorticity at the two upper corners. As

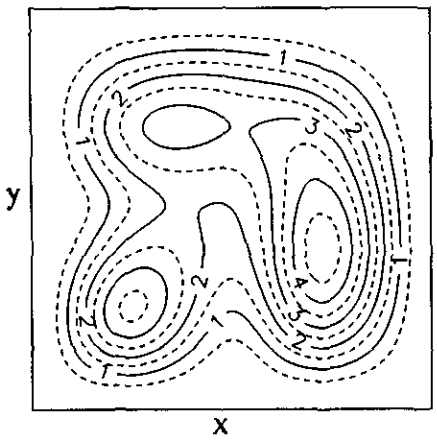

FIG. 4. Streamfunction distribution ($\psi \times 1000$) for Re = 1000, Example 1, Section 2.1.E.

a matter of fact, all problems where the boundary has corners (step contraction of the channel, cavity problem with smooth boundary conditions, etc.) have solutions which are either themselves locally unbounded or have derivatives which are unbounded. One can determine analytically (for the Stokes flow) that near a concave corner (corner angle 90°) vorticity $\zeta \sim r^{1.74}$ and thus its second derivative is infinite, while for the convex corner (corner angle 270°) vorticity is itself singular, i.e., $\zeta \sim r^{-0.46}$ [18]. Here r denotes radial distance from the corner. The standard test examples, as discussed above, might be acceptable for lower order methods. Highly accurate methods, such as the one studied here, do pick up all inconsistencies in the problem specification and this makes their rational testing extremely taxing.

Further, one should keep in mind that the computational test must have a non-trivial solution, so that $u\zeta_x + v\zeta_y$ does not vanish, while the boundaries are as simple as possible. These requirements are fulfilled, for example, by a channel flow driven by imposed periodic tangent and/or normal velocity distribution. No singularities are expected and the numerical solution remains smooth even for moderately high (10^3–10^4) Reynolds numbers.

2.1.E. *Numerical Examples*

EXAMPLE 1. The (artificial) flow in a unit square cavity is considered. The streamfunction (Fig. 4) is known and is described by formulas (2.23a)–(2.23c) with $\gamma_1 = 2 \cdot 10^{-3}$, $\gamma_2 = 10^{-3}$, $\delta = 0$. The flow is generated solely by the presence of a non-potential body force (2.2) with velocity vanishing on the boundary. There is no velocity scale and the Reynolds number denotes the reciprocal of kinematic viscosity. The tests were carrried out for $\mathrm{Re} = 1/\nu = 1000$. The Reynolds number calculated *a posteriori*, with the aid of maximum internal velocity $v_{\max}$, was much smaller, i.e., $\mathrm{Re}_{\max} = 1 \cdot v_{\max}/\nu \approx 22$. Solutions were sought on the sequence of grids (2.27)–(2.28). The main criteria for stopping of the Gauss–Seidel iterations were $\max|\zeta^{m+1} - \zeta^m| < 10^{-10}$, $\max|\psi^{m+1} - \psi^m| < 10^{-10}$, where max was taken over the whole grid and m denoted iteration number. The error was defined as a maximum (over the whole grid) of the difference between the calculated $\zeta(h)$ and the exact $\zeta(0)$ values of vorticity, i.e.,

$$E(h) = \max|\zeta(h) - \zeta(0)|. \tag{2.29}$$

Figure 5 shows results obtained with different versions of the algorithm. Second-order approximation of the field equations [1] and Jensen boundary condition formula (2.17) ($H^2FE + H^2BC$ on Fig. 5) yield second-order convergence of the error, as expected. The fourth-order discretization described in Secton 2.1.B was tested with various formulas for boundary vorticity (see all H^4FE in Fig. 5). Surprisingly, the second-order Jensen formula (2.17) (*J-BC*) delivers third-order accuracy while the original Dennis–Hudson formula (2.19) (*DH-BC*) gives only second-order. Both the third- (2.21) (H^3BC) and the fourth-order (2.20) (H^4BC) formulas give fourth-order convergence of the error and comparable absolute errors. Figure 6 shows that when second-order formulas are used for velocity evaluation, the accuracy of the whole algorithm deteriorates to second order, despite the fact that the field equations and boundary conditions are approximated with fourth-order accuracy. Figure 7 gives error estimates obtained without using exact solutions and compares them to those obtained using the exact solution. The estimates (Eqs. (2.25)–(2.26)) are

$$\tilde{E}(h_k) = \Delta(h_k)/(1 - q^\alpha) \approx 1.66\,\Delta(h_k),$$
$$q = h_k/h_{k+1} \approx 2^{-1/3},\ \alpha = 4, \tag{2.30}$$

where $\Delta(h_k) = \max|\zeta(h_k) - \zeta(h_{k+1})|$ and the max is taken over

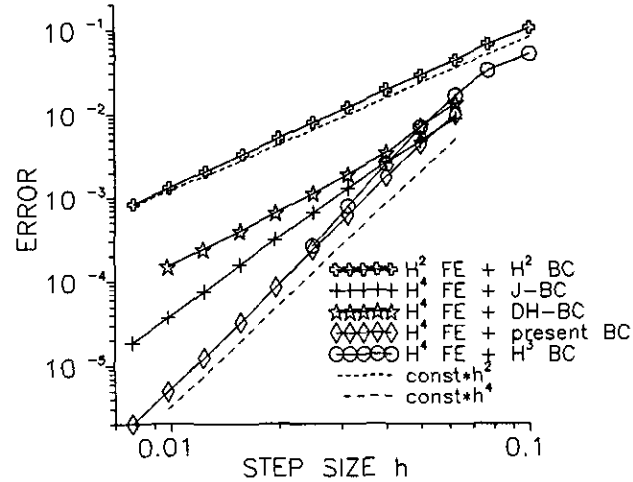

FIG. 5. Maximum error of vorticity (Eq. (2.29)) for different discretizations of the field equations (FE) and boundary conditions (BC). Calculations for Example 1, Section 2.1.E, Re = 1000.

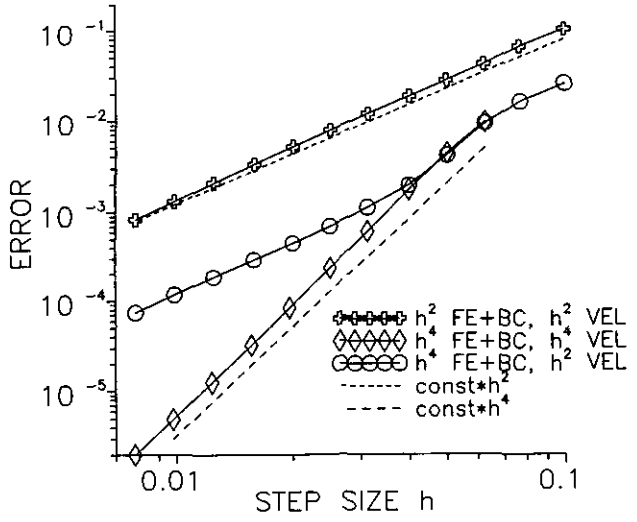

FIG. 6. Maximum error of vorticity (Eq. (2.29)) for different discretizations of velocity (VEL). Other conditions as in Fig. 5. See text for details.

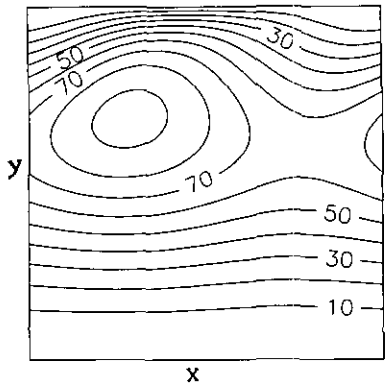

FIG. 8. Streamfunction distribution ($\psi \times 1000$), Re = 1000, Example 2, Section 2.1.E.

the boundary only. Since this error is almost equal to the exact error $E(h)$ given by (2.29) (where the max is taken over the whole grid), the results displayed in Fig. 7 show that the biggest local error occurs on the boundary and that $\zeta(h) - \zeta(0)$ converges monotonically as $h \to 0$.

EXAMPLE 2. In this example, we shall analyze the performance of the algorithm in the case of periodic channel flow, where for simplicity the period and the channel height are of equal length. The flow is generated by specifying tangential velocity distribution $v_\tau(x)$ at the upper wall in the form $v_\tau(x) = \sin^2(x/\pi)$, $x \in \langle 0, 1\rangle$, $v_{\tau\max} = 1$, which guarantees sufficient smoothness of the solution. The streamfunction vanishes on both walls and the Reynolds number is defined by $\mathrm{Re} = 1 \cdot v_{\tau\max}/\nu = 1/\nu$. The exact solution of this problem is not known. The numerical solution is calculated on the sequence of grids (2.27), (2.28), using the $H^4FE + H^4BC$ version of the algorithm. A typical pattern of streamfunction distribution is shown in Fig. 8. In the further discussion, $\zeta(h_k)$ denotes the solution obtained on kth grid, and $\zeta(0)$ denotes the unknown exact solution. Grid functions $\zeta(h_k)$ are extended onto the whole boundary using a fourth-order accurate spline function. The relative error $\Delta(h_k)$ between two approximate solutions is defined as

$$\Delta(h_k) = \max|\zeta(h_k) - \zeta(h_{k+1})|/\zeta_{\max}, \tag{2.31}$$

where max is taken over the whole boundary, and $\zeta_{\max}$ is the maximum absolute value of the vorticity (as found on the finest grid).

Figure 9 shows the distribution of the relative error $\Delta(h)$ as a function of step size h for different Reynolds numbers. Since

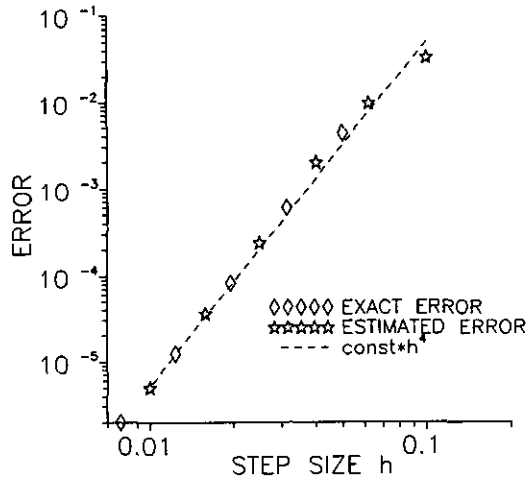

FIG. 7. Comparison of the exact (Eq. (2.29)) and estimated (Eq. (2.30)) errors of vorticity. Other conditions as in Fig. 5.

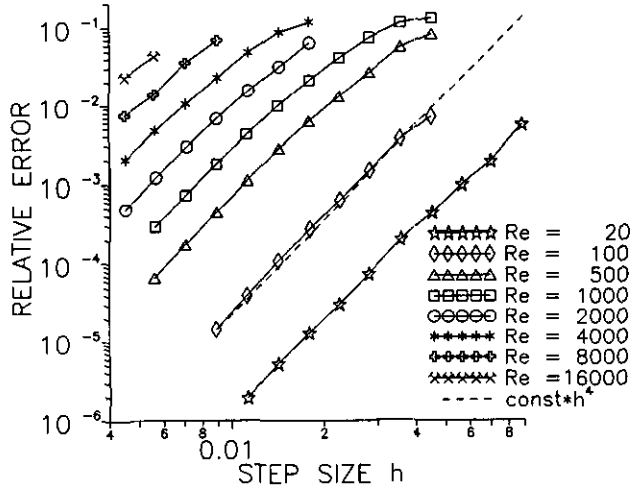

FIG. 9. Estimated relative error of vorticity (Eq. (2.31)), Example 2, Section 2.1.E.

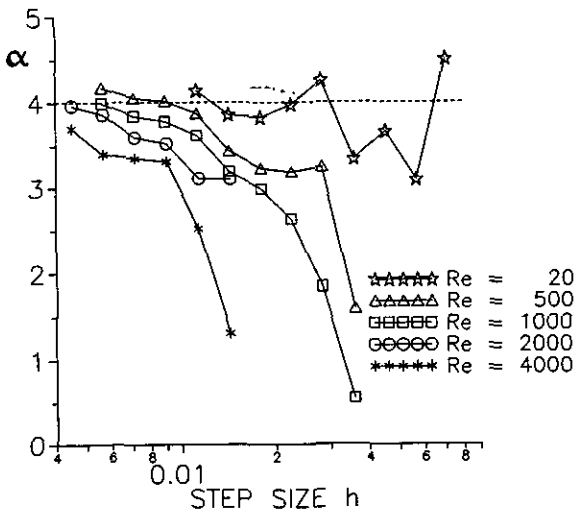

FIG. 10. Estimated order of discretization error (Eq. (2.32)), error ≈ const · h^{α}. Other conditions are as in Fig. 9.

the error $\Delta(h)$ is expected to be proportional to h^{α} we also calculate the exponent α (see Fig. 10) from

$$\alpha(h_k) = \ln(\Delta(h_k)/\Delta(h_{k-1})/\ln(h_k/h_{k-1}). \tag{2.32}$$

In all cases the fourth-order convergence (i.e., $\alpha \approx 4$) was obtained if the step size h was small enough. Analysis of $\Delta(h)$ in the limit of $h \to 0$ gives

$$\Delta(h) = b \cdot \mathrm{Re}^2 \cdot h^4 \quad (h \to 0), \tag{2.33}$$

where $b = 0.3 \pm 0.05$, regardless of the particular values of Re and h. It can be estimated using (2.25)–(2.26) that the true relative error $E(h) = \max |\zeta(h) - \zeta(0)|/\zeta_{\max}$ is not larger than

$$\tilde{E}(h) \approx 1.66\, \Delta(h) \sim \mathrm{Re}^2 \cdot h^4/2. \tag{2.34}$$

Thus, if accuracy $\tilde{E}(h)$ better than ε is desired, one must use a grid with step size

$$h \sim \mathrm{Re}^{-1/2} \cdot \varepsilon^{1/4} \tag{2.35}$$

or smaller. When h exceeds $\mathrm{Re}^{-1/2}$, the results become meaningless from the accuracy point of view and; in addition, the Gauss–Seidel iterations could become divergent. An increase of Re necessitates reduction of the underrelaxation parameters for vorticity and streamfunction, ($\omega \leq 0.1$ for Re > 4000) and introduction of underrelaxation of velocity. One should note that the condition $h < \mathrm{Re}^{-1/2}$ is by no means limited to our method or to the fourth-order methods, in general. In fact, the second-order method of Dennis and Hudson [1] can be interpreted as a conventional finite-difference scheme applied to the Navier–Stokes equations in which kinematic viscosity ν is distorted by a term proportional to h^2. One cannot expect results to be accurate unless h is (much) less than $\mathrm{Re}^{-1/2}$ (Re ~ $1/\nu$).

To compare efficiency of the fourth- and second-order methods, we repeated the same calculations using the fourth-order algorithm described in this paper and the second-order algorithm of Dennis and Hudson [1] with Jensens boundary formula (2.17). The solution obtained by the fourth-order method on the finest grid ($h = h_{\min}$) was used as a benchmark. The relative error $E_*^{(s)}(h)$ for different Reynolds numbers (see Fig. 11) was estimated as

$$E_*^{(s)}(h) = \max|\zeta^{(s)}(h) - \zeta^{(4)}(h_{\min})|/\zeta_{\max} + \tilde{E}(h_{\min}), \quad s = 2, 4, \tag{2.36}$$

where $\zeta^{(s)}(h)$ stands for vorticity obtained on the grid with the step h, using the method of order s ($s = 2$ or 4) and the maximum is taken over the boundary. The relative error $\tilde{E}(h_{\min})$ of the fourth-order solution on the finest grid ($h = h_{\min}$) was estimated by formula (2.34). In the calculations, $h_{\min} = \frac{1}{200}$ for Re = 500 and $h_{\min} = \frac{1}{250}$ for Re = 2000 and 4000. Analysis of the asymptotic behaviour ($h \to 0$) of $E_*^{(2)}(h)$ (second-order algorithm) gives $E_*^{(2)}(h) \sim \mathrm{Re} \cdot h^2$.

The cost of both methods can be compared by assuming, for example, that it is necessary to find a solution for Re = 4000 with 10% accuracy. The total cost T of the algorithm may be estimated as

$$T = N^2 \cdot It \cdot C, \tag{2.37}$$

where $N = 1/h + 1$ = number of grid points in one direction needed to obtain the desired accuracy, It = number of iterations to get convergence, and C = cost of a single iteration. Numerical

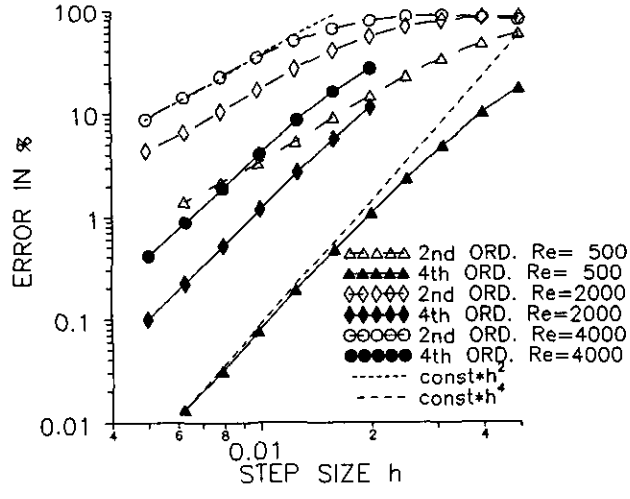

FIG. 11. Relative error of vorticity (Eq. (2.31))—comparison of the second- and fourth-order methods. Other conditions are as in Fig. 9. See text for details.

experiments give us in this case $N^{(2)}/N^{(4)} = 2.5$, $It^{(2)}/It^{(4)} = 3.5$, $C^{(2)}/C^{(4)} \sim 0.2 - 0.3$ (depending on the implementation), where superscripts $^{(2)}$ and $^{(4)}$ denote second- and fourth-order methods, respectively. The relative cost of both algorithms can be calculated from (2.37), i.e., $T^{(2)}/T^{(4)} \sim$ 4.5–6.5. Thus, the fourth-order algorithm does not only give more accurate results for the same grid as compared to the second-order algorithm, but it also requires significantly less computational labour to obtain results with the same absolute error.

2.2. Pressure Problem

2.2.A. *Preliminaries*

Suppose that velocity $\mathbf{V} = (u, v)$ and vorticity ζ form an exact solution of the flow problem (2.1a)–(2.1c). The corresponding pressure p can be obtained from the two-dimensional version of Eq. (B3) from Appendix A, i.e.,

$$\mathbf{grad}\, q = \mathbf{F}, \tag{2.38a}$$

where

$$q = (u^2 + v^2)/2 + p, \tag{2.38b}$$

$$\mathbf{F} = (F_1, F_2) = [v\zeta - \zeta_y/\mathrm{Re} + f_{e1}, -u\zeta + \zeta_x/\mathrm{Re} + f_{e2}] \tag{2.38c}$$

and subscripts x and y denote differentiation in the respective directions. Equation (2.38a) has a unique (within a constant) solution only if [19]

$$0 = \mathrm{curl}\, \mathbf{F} = \partial F_2/\partial x - \partial F_1/\partial y. \tag{2.39}$$

When $\mathbf{F}$ is defined by (2.38c), condition (2.39) is identically satisfied (it reduces to the vorticity transport equation). This shows that Equation (2.38a) is sufficient to determine the pressure field from the known velocity $\mathbf{V}$ and vorticity ζ fields.

Since it is more convenient to work with the second-order, rather than the first-order PDE, we take div of (2.38a) and obtain the Poisson equation for the total pressure q,

$$\Delta q = \zeta^2 + v\zeta_x - u\zeta_y + \mathrm{div}\, \mathbf{f}_e. \tag{2.40}$$

The boundary condition for q is obtained by projecting (2.38a) either onto the normal unit vector $\mathbf{n}$ at the boundary Γ,

$$\partial q/\partial \mathbf{n}|_\Gamma = \mathbf{F} \cdot \mathbf{n}, \tag{2.41}$$

or onto the tangent unit vector τ,

$$q|_\Gamma = \int \partial q/\partial \tau\, dt = \int \mathbf{F} \cdot \tau\, dt = g(t), \quad t \in \Gamma. \tag{2.42}$$

The last condition written for the horizontal (y = const) and vertical (x const) parts of the boundary takes the form

$$q|_\Gamma = \int (v\zeta - \zeta_y/\mathrm{Re} + \mathbf{f}_{e1})\, dx \qquad (y = \mathrm{const}), \tag{2.43}$$

$$q|_\Gamma = \int (-u\zeta + \zeta_x/\mathrm{Re} + \mathbf{f}_{e2})\, dy, \quad (x = \mathrm{const}). \tag{2.44}$$

Both the Neumann (2.41) and the Dirichlet (2.42) boundary conditions can form the basis for evaluation of the pressure from (2.40). Neumann conditions are used more often cf. [16, 20–22]. In the present work we implement Dirichlet conditions mainly because they are easier to incorporate in higher-order discretization schemes.

The equivalence between the original formulation (2.38a) and the formulation given by (2.40), (2.42) will be discussed later when considering the sensitivity of the Dirichlet problem (2.40), (2.42) to disturbances of the right-hand side of (2.38a). One should note that $g(t)$ is defined by (2.42) within an additive constant and, if the boundary consists of disjoint parts (the flow region is multiply connected), this constant may have different values on each part. Determination of these constants requires the solution of additional Dirichlet problems for the Laplace equation.

2.2.B. *Discretization of the Pressure Problem*

Solution of the pressure problem consists of three steps. First, the right-hand side of (2.40) and the boundary conditions (2.42) or (2.43)–(2.44) have to be evaluated. Here, the main difficulty consists in evaluation of the derivatives of vorticity. Second, the just evaluated boundary condition has to be corrected in order to guarantee a single-valued solution. Third, the Poisson equation for q with the corrected boundary condition has to be solved numerically.

We begin our discussion with the fourth-order formulas for the evaluation of ζ_x and ζ_y, that are required in (2.40)–(2.44). Use of the Taylor expansion gives

$$\zeta_{x0} = (\zeta_1 - \zeta_3)/2h - \zeta_{3x0}h^2/6 + O(h^4)$$
$$\zeta_{y0} = (\zeta_2 - \zeta_4)/2h - \zeta_{3y0}h^2/6 + O(h^4), \tag{2.45a}$$

where the index notation is explained in Fig. 1. The higher derivatives of vorticity are expressed as

$$\zeta_{3x0} = (\Delta\zeta)_{x0} - \zeta_{x2y0}, \quad \zeta_{3y0} = (\Delta\zeta)_{y0} - \zeta_{2xy0} \tag{2.45b}$$

and the required derivatives of the Laplacian of vorticity are determined from derivatives of the vorticity transport equation (2.1a), i.e.,

$$(\Delta\zeta)_{x0} = \mathrm{Re} \cdot [u_{x0}\zeta_{x0} + u_0\zeta_{xx0} + v_{x0}\zeta_{y0} + v_0\zeta_{yx0} - (\mathrm{curl}\, \mathbf{F}_e)_{x0}]$$
$$(\Delta\zeta)_{y0} = \mathrm{Re} \cdot [u_{y0}\zeta_{x0} + u_0\zeta_{yx0} + v_{y0}\zeta_{y0} + v_0\zeta_{yy0} - (\mathrm{curl}\, \mathbf{F}_e)_{y0}]. \tag{2.46}$$

In the above, ζ_{x2y0} denotes $\partial^3\zeta/\partial \times \partial y^2$ at point "0" (Fig. 1) and other indices have similar meanings. All the derivatives in (2.46) and ζ_{x2y0}, ζ_{2xy0} in (2.45) should be approximated with the usual second-order discretization formulas.

The fourth-order formulas for ζ_x and ζ_y at the boundary can be obtained in a similar manner. The required second normal derivative ζ_{nn0} has to be known with only second-order accuracy and can be calculated either by using a non-compact finite difference formula $\zeta_{nn0} = (2\zeta_0 - 5\zeta_1 + 6\zeta_2 - 3\zeta_3)/h^2 + O(h^2)$ or by utilizing the vorticity transport equation in $\zeta_{nn0} = (\Delta\zeta)_0 - \zeta_{\tau\tau 0}$.

One should note that since in the calculations the vorticity transport equation (2.1a) is not enforced at the boundary (see Section 2.1.B), its repeated use in deriving discretization formulas that are used at the boundary may result in deterioration of the overall accuracy. This reduction is minor, as shown in Section 2.2.D.

We shall now discuss the appropriate evaluation of the boundary condition. The function $g(t)$ in (2.42) is calculated with fourth-order accuracy by analytically integrating the spline interpolant of $\mathbf{F} \cdot \tau$ with $\mathbf{F}$ defined by (2.38c). One should expect that because of discretization errors, the calculated $g(t)$ will not be a single-valued function; i.e., if the boundary forms a closed curve, $g(t)$ will not return to its original value after making a full circle. A non-zero coefficient of the form

$$\beta = g(T) - g(0) = \int_0^T \mathbf{F} \cdot \tau \, dt \tag{2.47}$$

provides a good measure of this nonuniqueness. Here t denotes arc length measured from a suitable reference point on the boundary Γ while T stands for the total length of the boundary curve. It is shown in the next section that when the actual (exact) pressure field is a single-valued function, the coefficient β is $O(h^4)$. If the pressure is not a single-valued function (cf. Section 2.2.D) the coefficient β = const + $O(h^4)$ where const = $O(1)$.

Construction of a properly posed boundary problem for q requires replacing $g(t)$ in (2.42) with $g_*(t)$, obtained by substracting a linear function from the calculated integral, i.e.,

$$g_*(t) := g(t) - \beta \cdot t/T = \int_0^t \mathbf{F} \cdot \tau \, dt - \beta \cdot t/T, \quad t \in \langle 0, T \rangle, \tag{2.47a}$$

so that $g_*(T) = g_*(0) = 0$. Use of $g_*(t)$, rather than $g(t)$ in (2.42) is necessary in all cases, even if $\beta \sim O(h^4)$.

The boundary problem for the single-valued q_* part of q has the form

$$\Delta q_* = R \equiv \zeta^2 + v \cdot \zeta_x - u \cdot \zeta_y + \operatorname{div} \mathbf{f}_\varepsilon \tag{2.48a}$$

$$q_*|_\Gamma = g_*(t) \tag{2.48b}$$

and is discretized using the general method described in Section 2.1.B with $\phi \equiv q$, $\bar{u} = \bar{v} \equiv 0$. The resulting system of linear algebraic equations is solved by using relaxation procedure (2.11). The numerical cost of obtaining the pressure is almost negligible in comparison with the cost of determination of the flow field.

In case β = const + $O(h^4)$, first the single-valued part of q is found from (2.48) and then a simple linear solution is added in order to satisfy the original equation (2.38a). The details are described in Section 2.2.D.

2.2.C. *Error Analysis*

The first-order equation for q (2.38a) was replaced, for the purpose of calculation, by a Poisson equation with the corrected boundary condition (2.48). The boundary correction may seem somewhat arbitrary and thus it is necessary to show that the numerical solution of (2.48) does form a fourth-order approximation of the exact solution of (2.38a).

It is assumed for the purpose of further discussion that the flow region Ω is simply connected, the pressure is single-valued, and all the functions involved are sufficiently regular. We use a tilde to indicate the calculated values as opposed to the exact values.

The reader should recall that the vorticity and velocity fields that are available do not satisfy the exact vorticity transport equation but only its discretized analog. Thus the right-hand side of (2.38a) (defined by (2.38c)) can be represented as

$$\tilde{\mathbf{F}} = \mathbf{grad}\, q + \varepsilon \cdot \mathbf{F}_\varepsilon, \tag{2.49}$$

where q denotes the exact solution of (2.38a) and $\varepsilon = O(h^4)$ is a measure of the magnitude of error $\mathbf{F}_\varepsilon$, made in the evaluation of $\mathbf{F}$. This error arises due to the use of approximate (i.e., calculated numerically), rather then exact, values of vorticity and velocity and due to the approximate differentiation of the vorticity. The error can be divided, without loss of generality, into potential and solenoidal components, i.e., as $\mathbf{F}_\varepsilon = \mathbf{grad}\, \phi_1 + \mathbf{curl}\, \phi_2$ [19]. This shows that $\tilde{\mathbf{F}}$ does not satisfy condition (2.39) and that the vector equation (2.38a) with $\tilde{\mathbf{F}}$ on the right-hand side has no solution. However, Eq. (2.48) rewritten in terms of $\tilde{\mathbf{F}}$, rather than the original $\mathbf{F}$, does have a unique solution $\tilde{q}_*$. The difference between q and $\tilde{q}_*$ is estimated below.

The right-hand side of the Poisson equation (2.48a) was obtained analytically from $\mathbf{F}$ and can be written as

$$\tilde{R} = R + O(h^4), \tag{2.50}$$

where the error term results from numerical evaluation of the derivatives of vorticity in (2.48) and is not directly related to $\mathbf{F}_\varepsilon$. The boundary function $\tilde{g}_*(t)$ obtained from (2.47a) can be expressed as

$$\begin{aligned}\tilde{g}_*(t) &= \int_0^t \tilde{\mathbf{F}} \cdot \tau\, dt - \beta \cdot t/T = g(t) - g(0) \\ &\quad + \varepsilon \cdot \int_0^t \mathbf{F}_* \cdot \tau\, dt - \beta \cdot t/T \\ &= g(t) - g(0) + \varepsilon[\phi_1(t) - \phi_1(0) \\ &\quad - \int_0^t \partial\phi_2/\partial\mathbf{n}\, dt] - \beta \cdot t/T + O(h^4),\end{aligned} \tag{2.51}$$

where the error term results from the numerical integration procedure and the evaluation of constant β gives

$$\beta = \int_0^T \tilde{\mathbf{F}} \cdot \tau\, dt = \varepsilon \cdot \int_0^T \partial\phi_2/\partial\mathbf{n}\, dt. \tag{2.52}$$

One should note that β does not vanish due to the fact that curl $\mathbf{F}_\varepsilon \neq 0$. Since $\varepsilon = O(h^4)$, the difference between q and $\tilde{q}_*$ (2.49)(2.52) satisfies the following:

$$\begin{aligned}\Delta(\tilde{q}_* - q) &= \tilde{R} - R = O(h^4) && \text{on } \Omega \\ (\tilde{q}_* - q) &= \tilde{g}_* - g_* = -g(0) + O(h^4) && \text{on } \Gamma.\end{aligned} \tag{2.53}$$

Since the Dirichlet problem for the Poisson equation depends continuously upon data [19], the solution of (2.53) can be represented as

$$\tilde{q}_* - q = \text{const} + O(h^4) \quad \text{on } \Omega. \tag{2.54}$$

This shows that $\tilde{q}_*$ is, within an additive constant, a representation of q with accuracy $O(h^4)$.

The formal equivalence between the first-order equation (2.38a) and the problem (2.48) can be proven using similar arguments by taking $\varepsilon = 0$ and neglecting approximation errors. One should keep in mind that when Ω is not simply connected the strict equivalence does not hold and additional Dirichlet harmonic problems have to be solved (cf. Section 2.2.B). This is so because the harmonic problem $\Delta w = 0$ with boundary condition $\partial w/\partial\tau = 0$ on Γ admits solutions other than constants, if Ω is multiply connected.

2.2.D. *Numerical Testing of the Pressure Solution*

The artificial solution described in Section 2.1.D cannot be extended to pressure calculations because neither the fictitious body force $\mathbf{f}_e$ needed to calculate $\mathbf{F}$ is available, nor can Eq. (2.38a) be solved analytically. Our testing is therefore limited to grid convergence studies only. Below we describe results of testing in the case of test example 2 from Section 2.1.E.

The test problem, i.e., periodic flow in a channel, has two important features. The first one is that the flow region is double-connected (the boundary Γ consists of two disjoint parts Γ_1 ($y = 0$) and Γ_2 ($y = 1$)). The second one is that, while the flow itself is periodic in the x-direction, the pressure may contain a linear component (similarly as in the case of Poiseuille flow). Indeed, with the imposed mass flux equal to zero ($\psi = 0$ on both walls), a negative pressure gradient in the x-direction is required in order to balance the driving force resulting from the presence of the positive tangent velocity at the upper wall. Because of that, the solution to the pressure problem has to be represented as

$$q = q_* + A \cdot x + B \cdot y, \tag{2.55}$$

where q_* denotes the periodic solution of (2.48a), such that $q_*(0, 0) = q_*(0, 1) = 0$. Boundary condition (2.48b) simplifies to the form

$$\begin{aligned}q_*(x, y) &= -\left(\int_0^x \zeta_y\, dt - x \int_0^1 \zeta_y\, dt\right)\Big/ \text{Re}, \\ y &= 0 \text{ or } 1, x \in \langle 0,1\rangle\end{aligned} \tag{2.56}$$

because the normal velocity v on Γ_1 and Γ_2 vanishes. The coefficients A and B in (2.55) were calculated using (2.38), i.e.,

$$\begin{aligned}A &= q(1, 0) - q(0, 0) = \int_0^1 q_x(x, 0)\, dx \\ &= -\int_0^1 \zeta_y/\text{Re}\, dx, \quad y \equiv 0\end{aligned} \tag{2.57}$$

$$\begin{aligned}B &= q(0, 1) - q(0, 0) = \int_0^1 q_y(0, y)\, dy \\ &= \int_0^1 -u\zeta + \zeta_x/\text{Re}\, dy, \quad x \equiv 0.\end{aligned} \tag{2.58}$$

The integrals (2.56)–(2.58) were evaluated numerically using the method described in Section 2.2.B. Pressure calculations were carried out on a sequence of grids (2.27)–(2.28). The numerical results $q_*(h_k)$ were suitably extended onto the whole boundary, using fourth-order accurate spline interpolation. The relative error $\Delta(h_k)$ between two approximate solutions was defined as

$$\Delta(h_k) = \max|q_*(h_k) - q_*(h_{k+1})|/q_{*\max}, \tag{2.59}$$

where max was taken over Γ_1 and Γ_2 and $q_{*\max}$ was the maximum absolute value of the total pressure q_* (as found on the finest grid).

Figure 12 shows the relative error $\Delta(h)$ plotted as a function of the step size h, for different Reynolds numbers. It can be seen that the estimated order of accuracy (exponent α in Fig. 13) approaches the expected value of $\alpha = 4$ in the limit as $h \to 0$ but at a slower rate then in the case of vorticity (see Fig. 10). This is so because the vorticity transport equation used in deriving boundary conditions was not enforced at the boundary as explained in Section 2.2.C. The error of pressure $\Delta(h_k)$ was found to be dominated by the error associated with evaluation of the normal derivative of vorticity on Γ_2 (upper boundary). The same error analysis was performed for the coefficients A and B calculated from (2.57), (2.58) and a similar pattern of convergence leading to $\alpha = 4$ as $h \to 0$ was obtained.

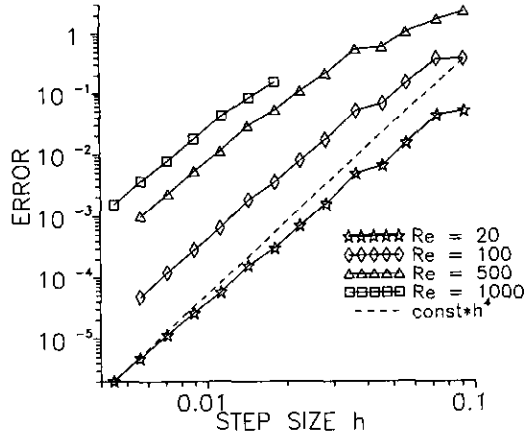

FIG. 12. Estimated relative error of the total pressure q_* (Eq. (2.59)). Other conditions are as in Fig. 9.

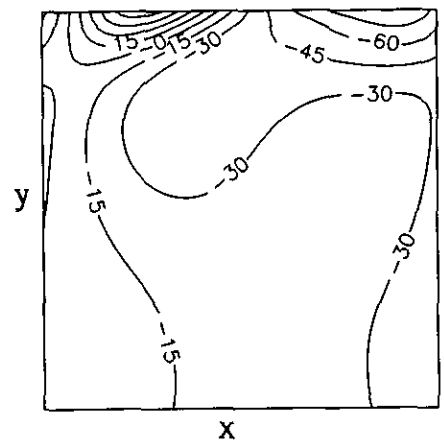

FIG. 14. Pressure distribution $p = q_* + Ax + By + (u^2 + v^2)/2$ (×1000) for Re = 100, Section 2.2.D.

Figure 14 shows topology of pressure distribution for Re = 100. This nonperiodic pressure field was calculated (cf. (2.38b) and (2.55)) from the formula $p = (u^2 + v^2)/2 + (q_* + Ax + By)$, where $A = -0.035677$, $B = -0.034427$. The pressure was normalized by setting it to zero at the lower left corner.

3. MULTIDOMAIN (MULTIPROCESSOR) IMPLEMENTATION

3.1. Preliminaries

The domain decomposition methodology consists in dividing the computational domain into overlapping subdomains and solving the original problem on each subdomain separately, with the appropriate transfer of information between the neighbouring subdomains. Use of this methodology offers two advantages. First a tremendous acceleration of calculations is possible, because each subdomain can be served by a different processor, with all the processors working concurrently. Second, modelling of complicated geometries can be simplified by breaking them into a union of geometrically (or topologically) simple subdomains.

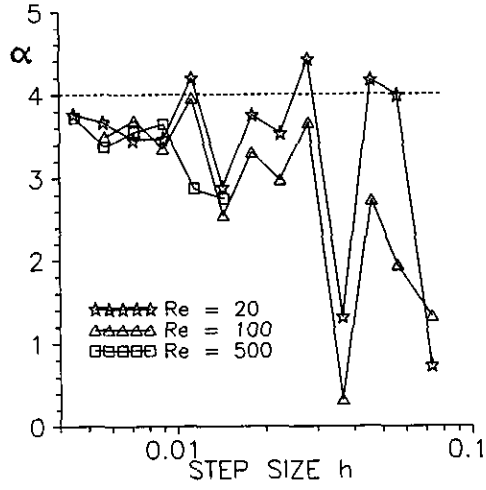

FIG. 13. Estimated order of discretization error of the total pressure q_*, error = const · h^α. Other conditions are as in Fig. 10.

Success of domain decomposition in accelerating the overall calculations depends strongly on the configuration of subdomains and on the strategy for information transfer between them. The questions of practical interest are (i) how the size δ of the overlap between subdomains affects the convergence speed, (ii) how often information between various subdomains should be exchanged, and (iii) what type of information should be transferred between the subdomains.

The general theory of the domain decomposition method for a second-order, linear elliptic PDE (Lions in [11]) states that this method is geometrically convergent, provided that the overlapping is uniform and the Dirichlet boundary information is transferred. The convergence rate is equal to

$$1 - C \cdot \delta \quad (\delta \ll 1), \tag{3.1}$$

where δ characterizes size of the overlap between neighbouring subdomains and the positive constant C depends both on the spectrum of the elliptic operator in question and on the dimension of the problem.

The Navier–Stokes equations (2.1a)–(2.1c), written in terms of streamfunction ψ form a fourth-order, biharmonic-like equation. The existing single-domain algorithms assume that either ψ and $\partial\psi/\partial\mathbf{n}$ or ψ and $\zeta = -\Delta\psi$ are known at the boundary. It is not clear which of these quantities should be transferred between subdomains to guarantee convergence of the algorithm. We shall begin the analysis by considering at first a very simple one-dimensional model problem and then use these results as guidelines in constructing the algorithm for the complete Navier–Stokes equations.

3.2. The Model Problem

Consider a boundary value problem for the fourth-order, linear, ordinary differential equation in the form

$$\begin{aligned} w^{IV} &= 0 \quad \text{on } \Omega = \langle 0, 2\rangle \\ w(0) &= w'(0) = 0 \\ w(2) &= w'(2) = 0 \end{aligned} \tag{3.2}$$

which has a trivial solution. In the above, w and w'' can be viewed as representing respectively ψ and $\Delta\psi = -\zeta$ in (2.1a)–(2.1c) with Re = 0. Let the domain Ω be divided into two overlapping subdomains Ω_1 and Ω_2 defined as

$$\Omega_1 := \langle 0, 1 + \delta\rangle, \quad \Omega_2 := \langle 1 - \delta, 2\rangle, \quad 0 < \delta \le 1,$$

with δ denoting the overlap size. The solution algorithm considered here consists in solving alternatingly the following two problems:

I. $w_1^{IV} = 0 \quad \text{on } \Omega_1$

$$\begin{aligned} &w_1(0) = w_2'(0) = 0, \quad w_1(1+\delta) = w_2(1+\delta), \\ &w_1^{(s)}(1+\delta) = w_2^{(s)}(1+\delta) \end{aligned}$$

II. $w_2^{IV} = 0 \quad \text{on } \Omega_2$

$$\begin{aligned} &w_2(2) = w_1'(2) = 0, \quad w_2(1-\delta) = w_1(1-\delta), \\ &w_2^{(s)}(1-\delta) = w_1^{(s)}(1-\delta), \end{aligned} \tag{3.3}$$

starting with arbitrary non-zero values of w_1 and w_2. We shall consider three types of information transfer, i.e., $s = 1, 2, 3$ in boundary conditions in (3.3). In the first one, information about the solution function and its first derivative in the overlap region is transferred; in the second one information about the function and its second derivative is transferred; and in the third one the second derivative is replaced by the third derivative.

One can write two problems forming (3.3) in a unified form as

III. $w_0^{IV} = 0 \quad \text{on } \langle 0, 1 + \delta\rangle$

$$\begin{aligned} &w_0(0) = w_0'(0) = 0, \quad w_0(1+\delta) = \theta_1, \\ &w_0^{(s)}(1+\delta) = \theta_2, \end{aligned} \tag{3.4}$$

where

$$\theta_1 = w_0^{OLD}(1-\delta), \quad \theta_2 = (-1)^s \cdot w_0^{(s)OLD}(1-\delta) \tag{3.5}$$

and superscript ^{OLD} denotes previous iteration. Indeed, substituting $w_0(x) := w_1(x)$ into (3.4) in the odd iterations and $w_0(x) := w_2(2 - x)$ in the even iterations gives exactly the alternating system (3.3). The exact solution to the general problem (3.4) has the form

$$w_0(x) = x^2 \cdot (ax + b), \tag{3.6}$$

where constants a and b are chosen from boundary conditions at $x = 1 + \delta$. Since (3.6) is linear with respect to a and b, the dependence of $\boldsymbol{\Theta} = (\theta_1, \theta_1)$ on (a, b) is also linear. This permits us to write the following relation at point $x = 1 - \delta$ (where the transfer of information (3.5) takes place):

$$\begin{aligned} w_0(1-\delta) &= B_{11}\theta_1 + B_{12}\theta_2 \\ w_0^{(s)}(1-\delta) &= B_{21}\theta_1 + B_{22}\theta_2. \end{aligned} \tag{3.7}$$

In the above, B_{ij} are functions of δ alone (different for $s = 1, 2, 3$). Combining (3.7) and (3.5) we get the transfer relation $\boldsymbol{\Theta}^{NEW} = \mathbf{B}_* \cdot \boldsymbol{\Theta}^{OLD}$ with matrix $\mathbf{B}_*$ in the form

$$\mathbf{B}_* = \begin{bmatrix} B_{11} & B_{12} \\ (-1)^s B_{21} & (-1)^s B_{22} \end{bmatrix}. \tag{3.8}$$

The convergence rate of the domain decomposition algorithm (3.3) is fully determined by the modulus of the largest eigenvalue $|\lambda_{max}|$ of the matrix $\mathbf{B}_*$. If this modulus is less than one, the convergence is geometrical; otherwise the algorithm is not, in general, convergent. The number of iterations N that are necessary to reach the prescribed accuracy can be estimated as $N \sim -1/\log|\lambda_{max}|$. Matrix $\mathbf{B}_*$ and its eigenvalues can be determined analytically. Details of this derivation are omitted due to its length. Table I gives expressions for $|\lambda_{max}|$ when δ is small for $s = 1, 2, 3$ and the corresponding number of iterations N. It is worth nothing that $\mathbf{B}_*$ possesses two distinct real eigenvalues for $s = 1, 3$ and complex eigenvalues for $s = 2$.

Analysis of the results given in Table I shows that the algorithm is practical for $s = 2$, i.e., when values of the function and its second derivative are transferred between subdomains. The convergence rate is similar to the one given by (3.1). In contrast, the first case ($s = 1$) requires an enormous number

TABLE I

Estimated Convergence Rates for the Domain Decomposition Algorithm for the Model Problem of Section 3.2

	Quantities transferred		
	w, $\partial w/\partial x$ $(s = 1)$	w, $\partial^2 w/\partial x^2$ $(s = 2)$	w, $\partial^3 w/\partial x^3$ $(s = 3)$
$\lvert\lambda_{max}\rvert$	$1 - 8 \cdot \delta^3$	$1 - 3 \cdot \delta$	1
$N \sim$	$1/8 \cdot \delta^{-3}$	$1/3 \cdot \delta^{-1}$	∞

Note. λ_{max} = maximum eigenvalue of the transfer matrix $\mathbf{B}_*$; N = estimated number of iterations.

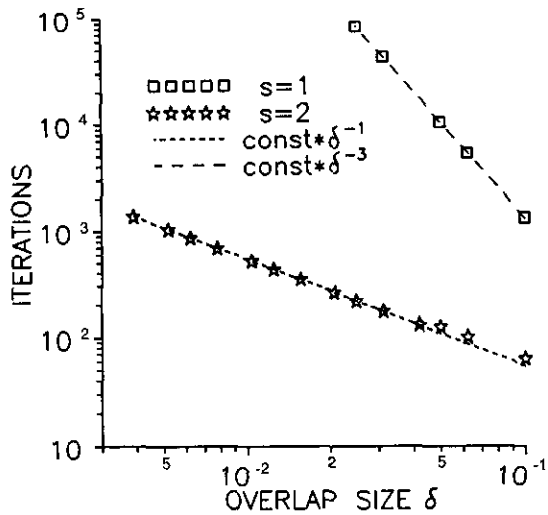

FIG. 15. Testing of convergence speed of the domain decomposition algorithm using different data transfer strategies for the model problem from Section 3.2.

of iteration and thus is impractical, while for $s = 3$ the algorithm is not convergent.

Analytical results discussed above were verified numerically for the discrete version of (3.4). The equation was discretized using standard second-order finite-difference formulas and its solution was sought using an algorithm equivalent to the one described in Section 2.1.A. The solution process consisted in splitting of (3.4) into two separate, second-order equations (the first one for w_0 and the second one for w_0''). The boundary condition for w_0'' was adjusted in the inner iteration loop using the boundary formula similar to the one described in Section 2.1.C. The information was transferred between subdomains only after the convergence of the inner iteration loop was achieved. The number of iterations of the domain decomposition algorithm (i.e., the number of information transfers) required to obtain accuracy of 10^{-7} is plotted in Fig. 15 as a function of the length δ of the overlap region. The results show full qualitative agreement with the analytical predictions given in Table I. In the case of $s = 3$ the algorithm was not convergent as predicted.

3.3. Characteristic Functions on Subdomains

There are various possible implementations of the domain decomposition method if the domain Ω is split into more than two subdomains, especially if there is multiple overlapping. The variant chosen here will be described by at first defining the characteristic functions on subdomains. These functions will be used in Section 3.4 to compose the complete solution on Ω from the solutions determined on each of the subdomains Ω_j.

Consider a bounded, connected open domain $\Omega \in \mathbb{R}^n$ ($n = 1, 2, 3$) that is split into K (connected) subdomains $\Omega_1, \ldots, \Omega_K$ (cf. Lions in [11]) such that $\Omega = \Omega_1 \cup \Omega_2 \cup \ldots \cup \Omega_k$. We denote by Γ and Γ_j the boundaries of Ω and Ω_j, respectively ($j = 1, 2, \ldots, K$). Let $\gamma_j = \Gamma_j \backslash \Gamma$ be the interface on which the information transfer to the jth subdomain takes place. Let A be an arbitrary point in Ω and its internal distance $d_j(A)$ from the interface γ_j be defined as

$$d_j(A) := \begin{cases} 0, & A \notin \Omega_j \\ \rho(A, \gamma_j), & A \in \Omega_j, \end{cases} \tag{3.9}$$

where $\rho(A, \gamma_j)$ stands for the distance between the point A and the interface γ_j. The largest internal distance from any interface γ_j is denoted by

$$d(A) := \max_{j=1,\ldots,K} d_j(A). \tag{3.10}$$

The "smallest" $d(A)$ over the whole Ω defines the characteristic size δ of the overlap structure

$$\delta = \inf_{A \in \Omega} d(A) := \inf_{A \in \Omega} \max_{j=1,\ldots,K} d_j(A). \tag{3.11}$$

We consider only the case when δ is positive, which means that for every point $A \in \Omega$ the internal distance between this point and at least one of the interfaces γ_j ($j = 1, 2, \ldots, K$) is greater than or equal to δ. This implies uniform overlapping as defined by Lions in [11].

The characteristic function χ_j is defined by the formula

$$\chi_j(A) := d_j(A)/S, \quad S = \sum_{p=1}^{K} d_p(A), \; j = 1, 2, \ldots, K, \tag{3.12}$$

and has the following properties:

$$\text{(i)} \; 0 \le \chi_j(A) \le 1 \tag{3.13a}$$

$$\text{(ii)} \; \chi_j(A) \equiv 0 \quad \text{if } A \notin \Omega_j \tag{3.13b}$$

$$\text{(iii)} \; \chi_j(A) \equiv 1 \quad \text{if } A \in \Omega_j, A \notin \Omega_k \; (k \neq j) \tag{3.13c}$$

$$\text{(iv)} \; \sum_{j=1}^{K} \chi_j(A) \equiv 1 \quad \text{for every } A \in \Omega \tag{3.13d}$$

$$\text{(v)} \; |\nabla \chi_j| \le 2 \cdot (K + 1)/\delta, \quad \text{wherever } \nabla \chi_j \text{ exists.} \tag{3.13e}$$

To prove the last inequality observe that $\nabla \chi_j = (\nabla d_j/S \cdot \nabla S)/S$. Since $d/S \le 1$ and $S \ge d \ge \delta$ we obtain $|\nabla \chi_j| \le (|\nabla d_j| + \sum_{p=1}^{K} |\nabla d_p|)/\delta$. Consider now the sphere of radius $\delta_0 < \delta$ with the centre at the point A, such that every interface γ_p ($p = 1, \ldots, K$) lies outside this sphere (cf. (3.9)–(3.11)). For every point B of this sphere we have $d_q(A) - \rho(A, B) \le d_q(B) \le d_q(A) + \rho(A, B)$, $q = 1, 2, \ldots, K$, and $|d_q(A) - d_q(B)|/\rho(A, B) \le 1$, where $\rho(B, A)$ is a distance between A and B. Consequently, $|\nabla d_q|$, if it exists, is bounded; i.e., $|\nabla d_q| \le 2$.

Combining this with the estimate for $\nabla\chi_j$ given above, one obtains (3.13e).

Properties (3.13a)–(3.13e) are analogous to those which are necessary to guarantee geometric convergence of the domain decomposition algorithm in the case of a second-order elliptic PDE (Lions in [11]) with the convergence rate given by (3.1).

3.4. Transfer of Boundary Information

The characteristic functions $\chi_1, ..., \chi_K$ introduced by (3.12) are used here to describe data transfer between subdomains. One may note that when more than two subdomains overlap, the simple alternating algorithm of Section 3.2 does not apply.

Consider that $w_1, ..., w_K$ are known continuous functions and that each w_j is defined on the corresponding subdomain Ω_j (w_j represents the solution obtained on the jth subdomain). Let each w_j be extended to the whole Ω by any constant (e.g., $w_j(A) \equiv 0$ if $A \notin \Omega_j$). The combined function w can be defined as

$$w(A) := \sum_{j=1}^{K} \chi_j(A) w_j(A), \quad A \in \Omega, \tag{3.14}$$

where w represents the resulting solution on Ω. The function $w(A)$ is continuous and reduces to $w_j(A)$ if $A \in \Omega_j$ and $A \notin \Omega_k$ ($k \neq j$). This property stems directly from (3.13c). The new boundary condition for w_j ($j = 1, 2, ..., K$) on the interface Ω_j is obtained by taking the value of $w(A)$ on γ_j, i.e.,

$$w_j^{\text{NEW}}(A) = w(A), \quad A \in \gamma_j. \tag{3.15}$$

One may note that this new boundary condition for w_j is influenced only by the information from the neighbouring subdomains and not by the old boundary value of w_j. This is so because $\chi_j(A) \equiv 0$ if $A \in \gamma_j$. If the splitting into subdomains is such that only two subdomains are permitted to overlap, then procedure (3.14)–(3.15) resembles the one already described in Section 3.3.

3.5. Domain Decomposition Algorithm for the Navier–Stokes Equations

The algorithm consists of the following steps:

1. initialize boundary conditions on each interface γ_j (e.g., $\psi, \zeta \equiv 0$ on γ_j, $j = 1, ..., K$),

2. perform P iterations of algorithm decribed in Section 2.1.A to solve Navier–Stokes equations on each subdomain separately (these calculations can be carried out simultaneously on the different processors),

3. calculate the combined function ψ and ζ defined on whole Ω, using (3.14) and values of ψ_j and ζ_j determined in step 2 for all Ω_j ($j = 1, ..., K$),

4. use (3.15) to calculate the new boundary values of ψ_j and ζ_j on all interfaces γ_j ($j = 1, ..., K$),

5. check for convergence of ψ and ζ on interfaces; if there is no convergence return to step 2.

An algorithm that transfers information regarding the normal derivative of streamfunction $\partial\psi/\partial\mathbf{n}$, rather than vorticity ζ, can be constructed in a similar manner.

The domain decomposition method described in Section 3.1 required that the data be transferred between subdomains only after the exact solution on each subdomain was found (P very large). When this solution is to be found iteratively, it is beneficial to interchange the domain decomposition iterations with the equation solver iterations. Our experiments with a single processor computer show that the algorithm requires the least time when P is equal one. This estimate does not account for the time that is required to transfer information between different processors. This time may be of the same order of magnitude as the time necessary to perform single iteration of the flow field.

The structure of our algorithm (with ψ and ζ being transferred) is such that it solves two separate second-order elliptic PDEs using iteration processes that are almost completely decoupled. Indeed, one could use the domain decomposition method to solve Eqs. (2.4) and (2.5) separately. This explains why the algorithm has to fulfil the convergence conditions (due to domain decomposition) that are identical to those derived in context of second-order linear PDEs (cf. Section 3.3). If information about ψ and $\partial\psi/\partial\mathbf{n}$ are transferred between subdomains this separation property obviously does not hold.

3.6. Numerical Testing of the Algorithm

The algorithm described in Section 3.5 was tested using examples described in Section 2.1.E. Two types of splitting of the flow domain Ω were employed, i.e., splitting into a sequence of either vertical or horizontal rectangles (Fig. 16a), with not more than two subdomains overlapping, and splitting into a matrix-like structure of almost square subdomains (Fig. 16b), with up to four subdomains overlapping simultaneously. For periodic domains there was also an overlapping between subdomains that share the periodic boundary. The overlap between subdomains was measured by δ (3.11), which in this case was equal to half the width of the overlapping rectangle. The calculations were performed for different δ, including the smallest possible overlap $2\delta = h$, where h stands for the discretization step size. All calculations described here were done using a single-processor computer.

Two variants of the domain decomposition method were tested, the first one with ψ and $\partial\psi/\partial\mathbf{n}$ and the second one with ψ and ζ being transferred between subdomains. In the former case, the algorithm described in Section 3.5 did not converge (the reader may recall that the model problem of Section 3.2 predicted an extremely low convergence rate).

It is of interest to check whether domain decomposition can influence the effective accuracy of the discretization schemes.

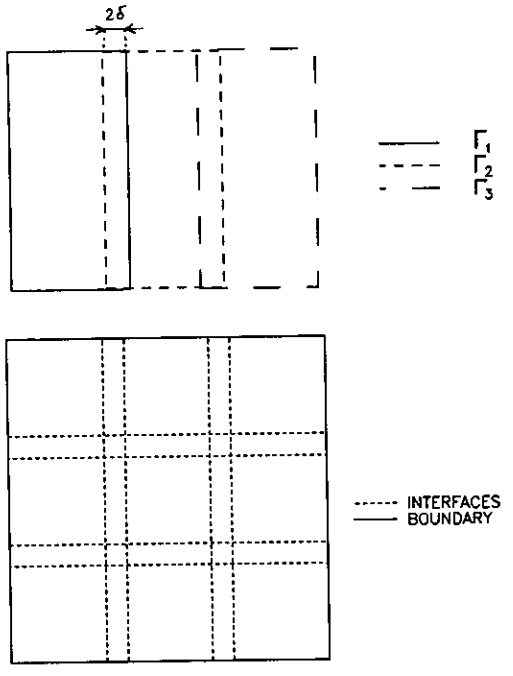

FIG. 16. Domain decomposition: (A) strips with single overlapping; (B) matrix-like structure with multiple overlapping.

Results of our numerical experiments show the absence of any such effect; i.e., in all cases the fourth-order accuracy of the solution was maintained.

The acceleration of calculations due to use of several processors, with each one serving a different subdomain, is best described by at first defining an acceleration factor in the form

$$A(K) = W(K)/W(1),$$

where $W(K)$ stands for the work effort associated with K processors (serving K subdomains) and $W(1)$ denotes the work effort of a single processor (single domain calculations). The acceleration factor cannot exceed the number of processors K and this defines the maximum theoretically possible acceleration of calculations. If $A(K)$ is less than one, application of the domain decomposition brings no benefits. The work effort of a hypothetical K-processor computer can be defined as

$$W(K) = [1 + OVL(\delta)] \cdot N(K)/K.$$

In the above, $N(K)$ stands for the number of iterations of the domain decomposition algorithm (Section 3.5), with each iteration consisting of a fixed number P of inner cycles of the Navier–Stokes solver. The work effort is reduced by a factor K because each of the K-processors will be serving $1/K$ part of the total grid, and it is increased by a factor of $(1 + OVL)$ because the total number of grid points increases due to overlap between the subgrids. Here, $OVL \sim \delta \ll 1$.

The acceleration factor calculated for the type of domain decomposition shown in Fig. 16b and with the smallest possible overlap $2\delta = h$ is displayed in Fig. 17 for various step sizes h. The results demonstrate that it is possible to come very close to the maximum theoretically possible acceleration rate and that the advantage of multiprocessing increases for increasing problem size. Similar results were also obtained for the type of domain decomposition shown in Fig. 16a.

4. CONCLUSIONS

A fourth-order finite-difference algorithm to solve Navier–Stokes equations in the streamfunction–vorticity formulation was developed. Various boundary formulas for vorticity were investigated, including new third- and fourth-order implicit formulas. An algorithm for calculation of pressure from the known velocity and vorticity fields was also presented.

Results of extensive numerical tests show that the algorithm does deliver fourth-order accuracy. The error has been estimated as being proportional to $\mathrm{Re}^2 \cdot h^4$. Comparison with a similar second-order algorithm shows that fourth-order algorithms require 4–6 times less computational time to obtain results with the same accuracy.

The domain decomposition method was incorporated into the algorithm in order to investigate performance gains resulting from the use of multiprocessor computers. It was determined that the algorithm converged only when information about the streamfunction and the vorticity was transferred between subdomains. The overall accuracy was not affected by use of the domain decomposition.

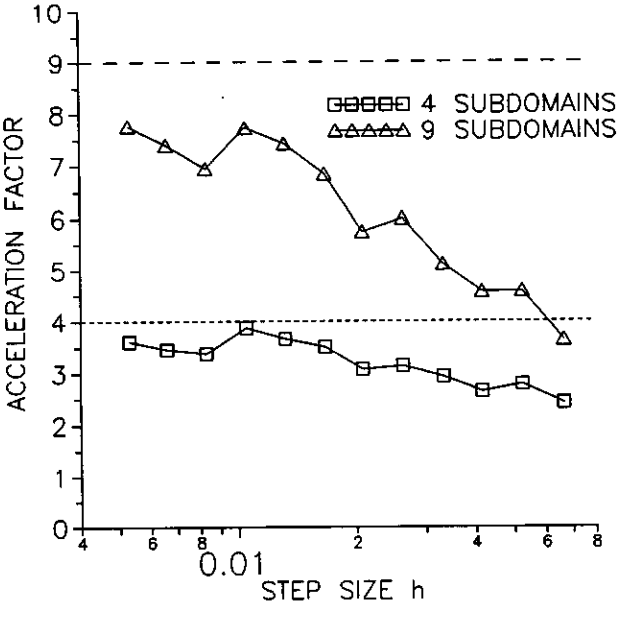

FIG. 17. Acceleration of calculations due to use of several processors, with each of them serving a different subdomain. Results for the flow described in Example 2 in Section 2.1.E with Re = 20.

Acceleration of the calculations very close to the maximum theoretically possible was observed. The advantage of multiprocessing increased with increasing the problem size.

APPENDIX A

The pressure equation can be derived from the momentum equation written in terms of primitive variables, i.e.,

$$(\mathbf{V} \cdot \nabla)\mathbf{V} = -\nabla p + \Delta\mathbf{V}/\mathrm{Re} + \mathbf{f}_e, \tag{A1}$$

where $\mathbf{V} = (u, v, w)$ denotes the velocity vector and $\mathbf{f}_e = (f_{e1}, f_{e2}, f_{e3})$ stands for an external body force vector. Use of the vector identities

$$(\mathbf{V} \cdot \nabla)\mathbf{V} = \mathbf{grad}\, V^2/2 - \mathbf{V} \times \nabla \times \mathbf{V}, \quad V^2 = \mathbf{V} \cdot \mathbf{V}$$
$$\Delta\mathbf{V} = \mathbf{grad}\,\mathrm{div}\,\mathbf{V} - \nabla \times \nabla \times \mathbf{V} \tag{A2}$$

and the zero velocity divergence assumption transforms (A1) (cf.[23, 22]) into

$$\mathbf{grad}\, q = \mathbf{F}, \tag{A3}$$

where $q = V^2/2 + p$, $\mathbf{F} = \mathbf{V} \times \nabla \times \mathbf{V} - \nabla \times \nabla \times \mathbf{V}/\mathrm{Re} + \mathbf{f}_e$.

REFERENCES

1. S. C. R. Dennis and J. D. Hudson, "A Difference Method for Solving the Navier–Stokes Equations," in *Proceedings, First International Conference on Numerical Methods in Laminar and Turbulent Flow, Swansea, United Kingdom, 1978,* edited by C. Taylor *et al.* (Pentech, London, 1978), p. 69.
2. R. K. Rubin and P. K. Khosla, *J. Comput. Phys.* **24,** 217 (1977).
3. R. S. Hirsh, *J. Comput. Phys.* **19,** 90 (1979).
4. S. C. R. Dennis and J. D. Hudson, *J. Comput. Phys.* **85,** 390 (1989).
5. M. M. Gupta, R. P. Manohar, and J. W. Stephenson, *Int. J. Numer. Methods Fluids* **4,** 641 (1984).
6. M. M. Gupta, *J. Comput. Phys.* **93,** 343 (1991).
7. M. M. Gupta and R. P. Manohar, *J. Comput. Phys.* **31,** 265 (1979).
8. R. Glowinsky and O. Pironneau, *SIAM Rev.* **21,** 167 (1979).
9. S. C. R. Dennis and L. Quartapelle, *Int. J. Num. Methods Fluids* **9,** 871 (1989).
10. L. Quartapelle and F. Valz-Gris, *Int. J. Num. Methods Fluids* **1,** 129 (1981).
11. R. Glowinsky *et al.* (Eds.) *First International Symposium on Domain Decomposition Methods for Partial Differential Equations* (SIAM, Philadelphia, 1988).
12. T. F. Chan *et al.* (Eds.), *Domain Decomposition Methods* (SIAM, Philadelphia, 1989).
13. T. F. Chan (Ed.), *Proceedings, Third International Symposium on Domain Decomposition Methods for Partial Differential Equations* (SIAM, Philadelphia, 1990).
14. R. Glowinsky *et al.* (Eds.) *Fourth International Symposium on Domain Decomposition Methods for Partial Differential Equations* (SIAM, Philadelphia, 1991).
15. J. S. B. Gajjar, *Comput. Phys. Commun.* **37,** 303 (1985).
16. P. J. Roache, *Computational Fluid Dynamics* (Hermosa, Albuquerque, NM, 1976).
17. D. F. Roscoe, *J. Inst. Math. Appl.* **16,** 291 (1975).
18. J. M. Floryan and L. Czechowski, Research Report ESFD-2/93, Dept. of Mechanical Engineering, University of Western Ontario, Canada, 1993 (unpublished).
19. V. Girault and P.-A. Raviart, *Finite Element Approximation of the Navier–Stokes Equations* (Springer-Verlag, Berlin/Heidelberg/New York, 1981).
20. S. A. Orszag, M. Israeli, and M. O. Deville, *J. Sci. Comput.* **1,** 75 (1986).
21. P. M. Gresho and R. L. Sani, *Int. J. Num. Methods Fluids* **7,** 1111 (1978).
22. M. O. Gunzburger, *Finite Element Methods for Viscous Incompressible Flows* (Academic Press, San Diego, 1989).
23. J. Rokicki and A. Styczek, *Arch. Budowy Maszyn* **39,** Z4, 351 (1992).