JOURNAL OF COMPUTATIONAL PHYSICS 96, 71–98 (1991)

Iterative Solution of Navier–Stokes Dual Variable Difference Equations*

GEORGE MESINA† AND CHARLES HALL

Institute for Computational Mathematics and Applications, University of Pittsburgh, Pittsburgh, Pennsylvania 15260

Received June 26, 1989; revised March 23, 1990

The dual transformation applied to implicit finite difference approximations of the Navier–Stokes equations reduces the number of unknowns by a factor of three, removes the pressures from the discrete equations and produces velocities which satisfy the discrete continuity equation exactly. New iterative methods for the solution of the unsymmetrical dual variable system are developed and are proven to converge for a large class of problems. These iterative methods involve a sequence of discrete Laplacian systems whose solutions converge to the solution of the dual variable system. They take advantage of the special structure of the dual variable coefficient matrix, are very fast compared to the direct methods currently used, are less memory intensive and can be more easily vectorized and parallelized. © 1991 Academic Press, Inc.

1. INTRODUCTION

The dual variable method was introduced by Amit, Hall, and Porsching [2] for certain finite difference schemes applied to the Navier–Stokes equations. This method eliminates pressure from the computations and, for 2D problems, reduces the number of active variables in the system to roughly one-third the original number. The dual variable method has been extended to finite element analyses [19], to three dimensions [25], and to compressible fluid problems [7]. It has been successfully used by a variety of authors, for example, to model cavity flows [6, 12, 14, 26] and thermally driven flows in reactor components [10, 20]. A key element of the implementation of the dual variable method is the construction of a null basis, a subject which has received much attention recently both in fluid and structural mechanics [4, 5, 8, 15, 16, 22].

The direct solution of the dual variable system requires less storage and computation time than the discrete primitive system. Iterative methods are generally faster and less memory intensive than direct methods for problems for which they are applicable; however, the dual variable system has previously always been solved by direct techniques because convergence of even such standard iterative methods

* This work partially supported by Air Force Office of Scientific Research Grant AFOSR-88-0262.
† Current address: EG&G, Idaho National Engineering Laboratory, Idaho Falls, ID 83415.

as SOR could not be guaranteed except in some very restrictive cases. This paper presents new iterative methods, which can be proven to converge to the solution for a large class of problems. They take advantage of the special structure of the dual variable coefficient matrix, are very fast compared to the direct methods currently used, are less memory intensive and can be more easily vectorized and parallelized.

1.1. *The Primitive Equations*

The basic equations used to model fluid flow are the two-dimensional time-dependent Navier–Stokes equations. The equations are simplified by assuming incompressibility, and a constant viscosity. The equations are the *continuity equation* (conservation of mass):

$$\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}=0 \tag{1}$$

and the *momentum equations* (conservation of momentum):

$$\rho\left(\frac{\partial u}{\partial t}+u\frac{\partial u}{\partial x}+v\frac{\partial u}{\partial y}\right)=-\frac{\partial p}{\partial x}+\mu\nabla^2 u+f_1 u+\rho g\cos\alpha \tag{2}$$

$$\rho\left(\frac{\partial v}{\partial t}+u\frac{\partial v}{\partial x}+v\frac{\partial v}{\partial y}\right)=-\frac{\partial p}{\partial y}+\mu\nabla^2 v+f_2 v+\rho g\sin\alpha \tag{3}$$

where u is the horizontal component of velocity, v is the vertical component of velocity, p is the pressure, ρ is the density, μ is the coefficient of laminar viscosity, $g(\cos(\alpha), \sin(\alpha))$ is the gravitational vector, and f_1 and f_2 are position-dependent friction factors.

Equations, (1)–(3), hold over a finite time interval $0\leqslant t\leqslant T$ and on a rectangular domain, Ω, with boundary $\partial\Omega$. Internal blockages and non-rectangular domains were handled explicitly in [1, 2, 10, 18, 20]; however, they can also be handled by introducing resistance terms in the momentum equation through f_1 and f_2.

The initial values for u and v for the above equations are given by

$$\begin{aligned} u(x, y, 0) &= u_0(x, y)\\ v(x, y, 0) &= v_0(x, y). \end{aligned} \tag{4}$$

It is not necessary to specify the initial pressures since the dual variable transformation eliminates them from consideration.

The boundary conditions are specified on segments of the boundary $\partial\Omega$. On a given segment, one of the following hold: (i) pressure specified, (ii) normal and tangential velocity specified, or (iii) normal velocity specified to be zero and the normal derivative of tangential velocity across the boundary segment is zero. Condition (ii) corresponds to a non-slip wall when the values are taken to be zero, while (iii) corresponds to a free-slip wall.

We now consider the stream function transformation of Eqs. (1)–(3). Introducing the stream function, ψ, enables one to eliminate the continuity equation and solve the resulting momentum equation directly for the single variable ψ. The stream function ψ is defined such that

$$-\frac{\partial\psi}{\partial x}=\rho v \quad \text{and} \quad \frac{\partial\psi}{\partial y}=\rho u. \tag{5}$$

With this definition of ψ, the mass velocity vector $\mathbf{q}=(\rho u, \rho v)=(\partial\psi/\partial y, -\partial\psi/\partial x)$. Note that (5) is consistent with the continuity equation since

$$0=\frac{\partial}{\partial x}(\rho u)+\frac{\partial}{\partial y}(\rho v)=\frac{\partial}{\partial x}\frac{\partial\psi}{\partial y}-\frac{\partial}{\partial y}\frac{\partial\psi}{\partial x}.$$

Finally, taking the curl of the momentum equation yields

$$\frac{\partial\psi}{\partial y}\frac{\partial}{\partial x}(\nabla^2\psi)-\frac{\partial\psi}{\partial x}\frac{\partial}{\partial y}(\nabla^2\psi)-\nu\,\nabla^2(\nabla^2\psi)=\operatorname{curl}(\mathbf{F}), \tag{6}$$

where $\nu=\mu/\rho$ is the kinematic viscosity, $\mathbf{F}=(F_1, F_2)$ and the curl is defined by $\operatorname{curl}(h_1, h_2)=\partial h_2/\partial x-\partial h_1/\partial y$.

Equation (6) is a scalar equation in just one variable, ψ. Moreover, use of the stream function obviates the need for the continuity equation. Thus the system of three equations, (1)–(3) is reduced to a single equation. In the process, the pressures have been eliminated and the mass velocities can be recovered in a straightforward way from (5). There are, however, some disadvantages. First, Eq. (6) contains third- and fourth-order derivatives. The finite difference approximations to these terms produce a very complicated coefficient matrix which is far less sparse than those arising from (1)–(3). Also, in order to uniquely solve (6) there is a need to impose auxiliary boundary conditions for the stream function. Properly selecting and enforcing these boundary conditions is generally more difficult than handling the boundary conditions introduced for Eqs. (1)–(3).

1.2. *The Discrete Primitive Equations*

It is possible to parallel the transformation to stream functions in the discrete case. We start with the discrete formulation of the primitive variable equations.

Consider Ω to be a rectangle in the x', y'-plane given by $\Omega=[0, a]\times[0, b]$ for some a, $b>0$. We overlay it with a mesh of lines given by $x'=x'_i$, $1\leqslant i\leqslant n$ and $y'=y'_j$, $1\leqslant j\leqslant m$, where $0=x'_0<x'_1<\cdots<x'_n=a$, $0=y'_0<y'_1<\cdots<y'_m=b$. The center of each mesh box (flow cell) is called a *node* and is located at (x_i, y_j), where $x_i=(x'_i+x'_{i+1})/2$, and $y_j=(y'_j+y'_{j+1})/2$.

The MAC placement of variables [10, 21, 28] is used; see Fig. 1. The discrete pressures approximate the true pressure at each node (center of mesh box) while the true mass-velocities are approximated by the discrete mass-velocities at the

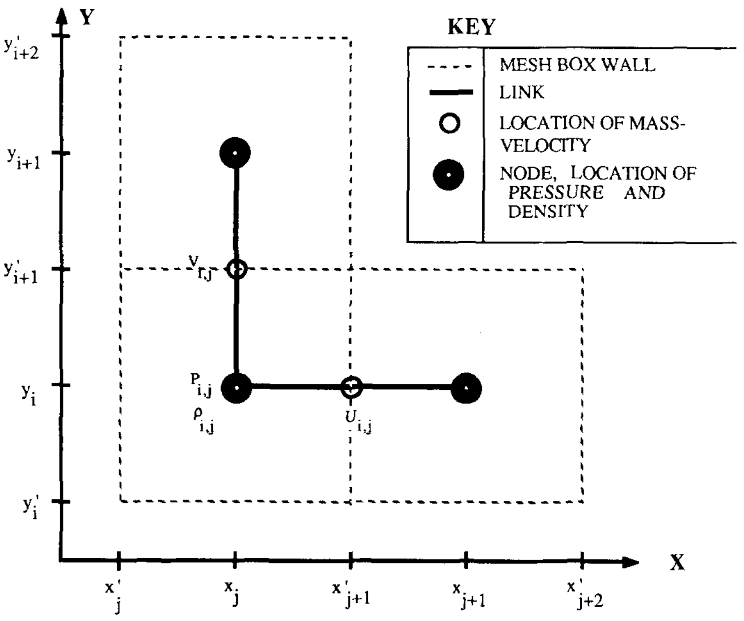

FIG. 1. Location of discrete variables.

midpoints of the mesh box sides. At time level k in cell centered at (x_i, y_j), let $P^k_{i,j} \approx p(x_i, y_j, k\,\Delta t)$, $U^k_{i,j} \approx \rho \cdot u(x'_{i+1}, y_j, k\,\Delta t)$, and $V^k_{i,j} \approx \rho \cdot v(x_i, y'_{j+1}, k\,\Delta t)$. Denote the discrete pressure vector at time level k by: $\mathbf{p}^k = (p^k_1, p^k_2, ..., p^k_N) = (P^k_{2,2}, P^k_{2,3}, ..., P^k_{n,m})$ and denote the discrete velocity vector by: $\mathbf{W}^k = (W^k_1, W^k_2, ..., W^k_L) = (U^k_{2,2}, U^k_{2,3}, ..., V^k_{n,m})$, where N is the number of unknown pressures and L is the number of unknown velocities. L is larger than N; for two-dimensional problems $L \approx 2N$.

The implicit difference equations are given in Appendix A. The continuity equation is discretized using centered differences. The momentum equation is discretized by using backward time differences for the temporal term, centered differences for the viscous and pressure gradient terms, and upwind differences for the convection term. The convection term is linearized by time-lagging the velocity. These equations are fully implicit and have no theoretical time-step restriction.

With these conventions, the equations can be written in matrix-vector form as

the *discrete continuity equation,*

$$A\mathbf{w}^{k+1} = \mathbf{s}^k \tag{7}$$

the *discrete momentum equation,*

$$Q^k\mathbf{w}^{k+1} = \Delta t A^{\mathrm{T}}\mathbf{p}^{k+1} + \mathbf{b}^k_1, \tag{8}$$

where $\mathbf{w}^{k+1} = D_1 \mathbf{W}^{k+1}$, the diagonal matrix D_1 is given in Appendix A and the vectors $\mathbf{s}^k$ and $\mathbf{b}_1^k$ contain the boundary information and source terms. The $N \times L$ matrix A is the *discrete divergence operator* acting on the space of velocity vectors and its transpose is a *discrete gradient* for the space of pressures [2, 26]. Q^k is the $L \times L$ matrix coefficients of the velocity terms of the momentum equation.

1.3. *Dual Variable Transformation*

The transformation from primitive variables to a stream function in the continuous case can be paralleled in the discrete case by constructing a discrete analog to the curl operator. The crucial properties of the curl that were used in the derivation of the stream function equation are that $\operatorname{div}(\mathbf{curl}(\mathbf{G})) = 0$ for all $\mathbf{G}$ in C^1 and that $\mathbf{curl}(\mathbf{grad}(g)) = 0$ for all g in C^1. For the discrete case, this translates into the requirement that a matrix, C, be constructed such that $AC = 0$ and $C^{\mathrm{T}} A^{\mathrm{T}} = 0$. Such a matrix always exists. It can be constructed by choosing its columns as a basis for the subspace of L-dimensional Euclidean space, R^L, orthogonal to the subspace spanned by the columns of A^{T}.

In practice, it is easy to construct both A and C by the use of graph theory; see [2, 3, 7, 10, 18, 20, 26]. Goodrich and Soh [14] have given an alternate, but equivalent, construction of the matrix C. For MAC-type meshes, A and C can be constructed so that $\operatorname{rank}(A) = N$ and C is an $L \times (L - N)$ matrix of rank $L - N$.

Let $\mathbf{v}_0^{k+1}$ be any vector in R^L which satisfies

$$A\mathbf{v}_0^{k+1} = \mathbf{s}^k. \tag{9}$$

The vector $\mathbf{v}_0^{k+1}$ is called a *particular solution* of the continuity equation. Note that for incompressible flows and time independent boundary conditions, $\mathbf{v}_0^{k+1}$ is independent of time and hence k. See [20] for thermally expandable problems where this is not the case. Since $A\mathbf{w}^{k+1} = \mathbf{s}^k$,

$$\mathbf{w}^{k+1} = \mathbf{z} + \mathbf{v}_0^k \tag{10}$$

is the general solution of the continuity equation, where $A\mathbf{z} = \mathbf{0}$. We now seek the unique vector $\mathbf{z}$ that satisfies the momentum equation as well. Since $\mathbf{z} \in \operatorname{Nullspace}(A) = \operatorname{Range}(C)$, $\mathbf{z} = C\mathbf{x}$ for some $\mathbf{x} \in R^{L-N}$. Next, we modify the momentum equation using (10) to obtain

$$Q^k \mathbf{z} = \Delta t \cdot A^{\mathrm{T}} \mathbf{p}^{k+1} + \mathbf{b}^k, \tag{11}$$

where $\mathbf{b}^k = \mathbf{b}_1^k - Q^k \mathbf{v}_0^k$.

Finally. take the discrete curl of Eq. (11) and replace $\mathbf{z}$ by $C\mathbf{x}$ to arrive at the *dual variable system*:

$$C^{\mathrm{T}} Q^k C \mathbf{x} = C^{\mathrm{T}} \mathbf{b}^k. \tag{12}$$

The vector $\mathbf{x}$ is called the *discrete* stream function or vector of *dual variables*.

Once the dual variable system (12) is solved for the dual variables, the velocities can be recovered at almost no cost from the formula:

$$\mathbf{w}^{k+1} = C\mathbf{x} + \mathbf{v}_0^k. \tag{13}$$

After the velocities have been recovered, the calculation of the next time level solution proceeds with the construction of a new particular solution from (9) (if needed), the construction of Q^{k+1}, the formation and solution of the new dual variable system, and the recovery of the new velocity field. The pressures are not needed for time-step advancement; they have ceased to be a problem variable. Also, the continuity equation (7) has been replaced by (9) which is decoupled from the momentum equation (8). Moreover, (9) requires very little computational work, reducing essentially to back-substitution of a sparse triangular system via the use of a spanning tree [2, 20].

There are many advantages to the dual variable method. The dual variable transformation does not place any special restrictions on the continuity equation. It does not require boundary conditions other than those needed for the discrete system of Eqs. (1)–(3). The continuity equation is satisfied exactly:

$$A\mathbf{w}^{k+1} = AC\mathbf{x} + A\mathbf{v}_0^k = \mathbf{s}^k. \tag{14}$$

Also, the discrete system (7)–(8) in three unknown vectors (pressure, horizontal, and vertical velocity) have been reduced to a system with just one unknown vector (discrete stream function). There are at most one-third as many discrete variables in (12) as were contained in (7)–(8); this sizable reduction in system size produces a large reduction in solution time.

The main disadvantage of the dual variable transformation is the structure of the coefficient matrix which we now examine.

The matrix Q^k in (8) results from a five-point stencil for both the horizontal and the vertical velocities; the five-point stencil arises from the usual centered difference approximation of the Laplacian and the upwind approximations to velocity gradient terms. As such, Q^k is block tridiagonal, and the diagonal blocks are themselves tridiagonal. Q^k is reducible since there is no connection between the equations of horizontal mass velocity and vertical mass velocity. Q^k is also an M-matrix (and hence non-singular) and is strictly diagonally dominant when a uniform mesh spacing is used [27]. However, Q^k is not symmetric due to the convective terms. The fact that Q^k is an M-matrix is enough to guarantee the convergence of Jacobi and SOR methods when applied to any system of equations whose coefficient matrix is Q^k [31]. Unfortunately, (8) also involves the unknown pressures.

The matrix C can be constructed from graph theory; first we describe the graph and its key features. Consider the nodes of the MAC grid to be the *nodes* of the graph. Connect adjacent nodes via directed horizontal and vertical line segments; the horizontal segments are directed to the right and the vertical ones are directed

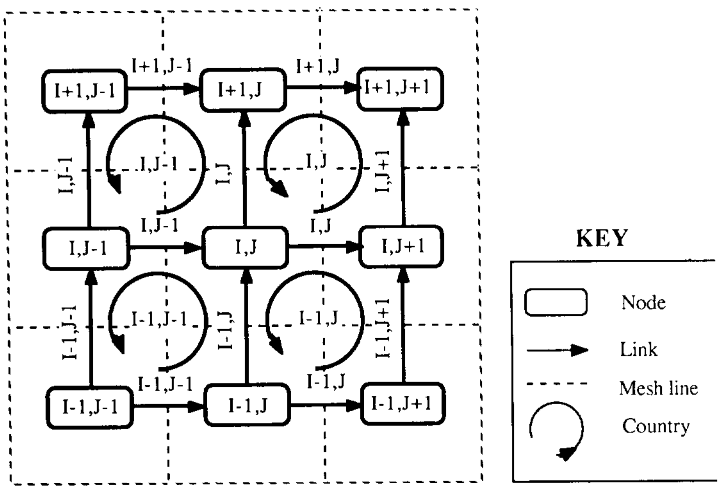

FIG. 2. Graph resulting from MAC grid.

upwards. These line segments are the *links* of the graph. Note that the links pass through the points where the unknown mass velocities are approximated. A *path* of a graph starts at any node and continues to any other node along a sequence of links regardless of their orientation but each of which is connected to the previous link at one of its endpoints and to the next link at the other. A *cycle* is a path which begins and ends at the same node and a *country* is a cycle that contains no nodes or links in its interior [3]. In our graph, most countries are cycles with four nodes and four links and have a rectangular shape (see Fig. 2). When a pressure is specified at a node on the boundary, the link which connects it to the adjacent node inside the region is associated with an unknown velocity. If more than one such pressure specification is made, a pseudo-country is formed; it is the shortest path between two specified pressure nodes. A pseudo-country is considered a special kind of country and may consist of less than four, four, or many more than four links and nodes (see, for example, [1, 2, 10, 18]).

The pth column of C, $c_p = (c_{1,p}, c_{2,p}, ..., c_{L-N,p})^T$, corresponds to the pth country, Ω_p, and is defined by traversing Ω_p in a counterclockwise manner. Define $c_{j,p} = 0$ if link j is not in Ω_p, $c_{j,p} = 1$ if link j is in Ω_p and its direction coincides with the direction it is traversed, and $c_{j,p} = -1$ if link j is in Ω_p and its direction is opposite to the direction of traverse. The fact that this matrix satisfies $AC = 0$ and has full rank is proven in [3].

The matrix Q^k represents the velocity-to-velocity momentum coupling or the link-to-link coupling; the non-zero entries of row i of Q^k are in the columns of the velocities coupled to w_i through the finite difference scheme. The matrix C represents the link-to-country coupling; the non-zero entries of row i of C are in the columns corresponding to the countries in whose boundary link i lies. Similarly,

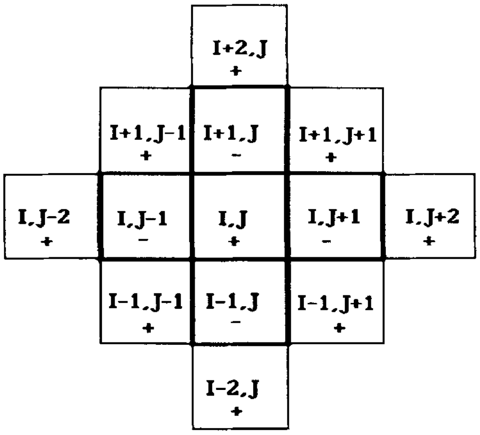

FIG. 3. The 13-point stencil for regular countries.

C^TQ^k is the country-to-momentum-neighbor coupling matrix; if $(C^TQ^k)_{i,j}$ is non-zero, then either link j lies in the boundary of country i or it is momentum-coupled to a link in the boundary of country i. C^TQ^kC is the country-to-country coupling matrix; if $(C^TQ^kC)_{i,j}$ is non-zero, then either country i and country j share a link, or there is a link in country i which is a momentum-neighbor to a link in country j. This last observation establishes that C^TQ^kC corresponds to a 13-point stencil for regular countries. In Fig. 3 the momentum couplings involved in the country couplings for country (I, J) are darkened and the sign of $(C^TQ^kC)_{i,j}$ is also given.

Even though Q^k is non-singular and C has full rank, it is not guaranteed that C^TQ^kC is non-singular. However, it was shown in [2] that whenever (7)–(8) has a unique solution, so too does (12) and vice versa.

The main disadvantage of the dual variable procedure is the structure of the coefficient matrix. Even for a flow region Ω with no blockages, the non-zero entries of C^TQ^kC follow the same pattern as the discrete biharmonic, but the matrix is neither symmetric, diagonally dominant, non an M-matrix. It has none of the well-known properties which guarantee convergence for the usual iterative techniques [27].

For regions Ω with blockages, there are two standard methods to handle the obstacles, both of which add further difficulties. The first is to remove the pressures and velocities within the obstacle from consideration as problem variables, [1, 2, 12]. This has the effect of producing a very large country corresponding to the smallest cycle wich surrounds the obstacle. It also destroys the banded structure, creating instead a coefficient matrix that is bordered-banded. An alternative is to place large resistances to the flow of the fluid through the blockages by use of friction factors f_i. This latter approach generally results in a very ill-conditioned dual variable system (12).

2. Solution Methods for the Dual Variable System

2.1. *Existing Methods*

Direct methods have been used exclusively to solve Eq. (12). The inherent limiting factor to a direct solution method is the large amount of memory required. For regions without blockages, Gaussian elimination (banded or profile) with partial pivoting appears to work well. However, for regions with blockages, the matrix is no longer banded, but border banded and the border may be quite large. Such systems can be solved very efficiently by special direct techniques [30] which both vectorize and parallelize. However, these techniques are inefficient for problems arising from regions without blockages. Some direct methods apply equally well to both flow region geometries. The dual variable system has been solved by full-matrix Gaussian elimination with partial pivoting [6, 25], but for problems of even moderate size the in-core storage and computation costs become prohibitive. Straightforward implementation of Gaussian elimination using peripheral storage reduces the in-core storage problem, but increases the runtime costs due to excessive communication between devices. To make effective use of peripheral storage, the frontal method [24] has been applied in the dual variable computer code DUVAL [1, 7, 10] and has been used to solve transient problems with up to 6400 flow cells on a CRAY/XMP. The reverse Cuthill–McKee algorithm was used to minimize the bandwidth of the banded portion of the dual variable system and optimize its profile. Moreover, with the frontal method, blockages add no further computational complexity to the solution procedure. That is, the border they generate is absorbed into the active matrix as the country associated with the blockage is encountered. Once the blockage is isolated behind the front, the border columns are eliminated. Thus the frontal method effectively handles the matrix as if it has no border. Finally, though frontal methods are slower in real time than in-core Gaussian elimination variants, because of the reliance on slower peripheral storage devices, for very large problems an in-core method is not a viable alternative, [1].

The anticipated payoff for using iterative methods instead of direct methods is reduced storage and reduced computation time. For example, for a square region with n dual variables in each row and column, C^TQ^kC has a bandwidth of $2n$. Thus to store the matrix, full-matrix Gaussian elimination uses $O(n^4)$ memory locations, a banded Gaussian elimination solver requires at least $O(4n^3)$ memory locations, a frontal solver such as implemented in DUVAL uses $O(16n^2)$ in computer memory plus pheripheral storage and a point iterative solver needs $O(13n^2)$.

Unfortunately, the dual variable system (12) does not possess the properties which guarantee convergence for the standard iterative methods. It is neither diagonally dominant nor an M-matrix; therefore the usual iterative methods, namely the point and block forms of Jacobi, Gauss–Seidel, and SOR, cannot be guaranteed to converge. The coefficient matrix is also neither positive definite symmetric nor symmetrizable (a matrix B is symmetrizable if there exists a positive definite matrix Z such that ZB is positive definite symmetric). Therefore the classical conjugate gradient (CG) method [23] cannot be applied to (12).

One can always use the normal equations,

$$(C^{\mathrm{T}}Q^kC)^{\mathrm{T}}\,(C^{\mathrm{T}}Q^kC)\mathbf{x} = (C^{\mathrm{T}}Q^kC)^{\mathrm{T}}\,C^{\mathrm{T}}\mathbf{b}, \tag{15}$$

which are automatically positive definite symmetric. There are several forms of CG, called CGN, which apply CG to (15) without actually forming the system. The main problem with using the normal equations is that the two-norm condition number of the normal equations is the square of the condition number of the original system (12); so that a large condition number in the dual variable system implies an enormous condition number in the normal equations. Ill-conditioning leads to extra iterations in any iterative method and the potential extreme ill-conditioning of the normal equations makes their direct use undesirable.

There are many generalizations of CG, among them, ORTHOMIN(k) can be shown to solve the system if the matrix Q is PDR (positive definite real) [27]. By use of a preconditioning matrix, the CG methods can be made to converge faster. However, the preconditioning matrix can be viewed as a matrix of parameters and a good choice of these parameters is not known a priori and may change with each time-step.

Multigrid is another popular iterative method for solving linear systems of equations arising from Navier–Stokes type equations including primitive systems such as (7)–(8). For a review of this literature, see [26]. This literature is not directly applicable here since it does not deal with the dual variable system, rather the primitive system or other formulations. It is very likely that the basic iterative schemes presented in the next section can be accelerated through the use of multigrid techniques. However, this topic is not pursued further in this paper.

Any iterative method applied to the primitive system (7)–(8) will yield a velocity field that only *approximately* satisfies the discrete continuity equation. In contrast, any iterative method applied to the dual variable formulation (12) yields velocity fields that satisfy the discrete continuity equation *exactly*. This is an inherent property of the dual variable formulation.

The object of the current research was to construct a robust iterative method for the dual variable system which could be shown to converge for a large class of problems.

2.2. *New Iterative Methods*

This section reports on several iterative methods for the solution of the dual variable method that were developed and analyzed in the dissertation [27]. The central idea behind these new iterative methods is to take advantage of the structure of $C^{\mathrm{T}}QC$ and a splitting $C^{\mathrm{T}}QC = P - R$ such that P can be inverted quickly and efficiently and such that the overall resulting iteration can be proven to converge for a large class of problems. This splitting leads to the iterative method

$$P\mathbf{x}^{k+1} = R\mathbf{x}^k + C^{\mathrm{T}}\mathbf{b}. \tag{16}$$

Equation (16) is called the *inner system*; it must be solved on every step of the iterative procedure.

To reduce and simplify notation, we establish the following definitions: Let

$$Q = Q_D - Q_L - Q_U, \tag{17}$$

where $Q_D := \mathrm{diag}(q_{11}, q_{22}, ..., q_{LL})$, Q_L is strictly lower triangular, and Q_U is strictly upper triangular. The matrix of off-diagonal terms, $F := Q_D - Q$ is split as $F = F_s + F_c$, where F_s contains only the off-diagonal stress terms of Q, and F_c contains the off-diagonal convective terms of Q. Note that $F = Q_L + Q_U$. With these definitions we derive several methods.

2.2.1. *The Stress-Convection Method*

Splitting the off-diagonal matrix, F, into a matrix containing the stress terms and a matrix containing the convective terms gives

$$C^T Q C\mathbf{x} = C^T(Q_D - F_S - F_c)\, C\mathbf{x} = C^T\mathbf{b}. \tag{18}$$

For uniform mesh-spacing, F_S is symmetric. Keeping the matrices which are symmetric, in the uniform mesh-spacing case, on the left produces the following iterative method:

$$C^T(Q_D - F_S)\, C\mathbf{x}^{n+1} = C^T F_c C\mathbf{x}^n + C^T\mathbf{b}. \tag{19}$$

This method is called the *stress-convection method* because all the viscous stress terms are in the splitting matrix and all the convective terms appear on the right side of Eq. (19). $C^T(Q_D - F_S)C$ is invertible [27]. The form of the iteration matrix G_S, given in Table I, now follows from inverting Eq. (19).

2.2.2. *The Transformed Jacobi Method*

This method comes from splitting the matrix Q as in (17) and then applying dual variable transformation:

$$C^T Q C\mathbf{x} = C^T(Q_D - Q_L - Q_U)\, C\mathbf{x} = C^T\mathbf{b}. \tag{20}$$

TABLE I

Iteration Matrices for the New Iterative Methods

Stress-convection	$G_S = [C^T(Q_D - F_S)C]^{-1}\, C^T F_c C$
Transformed Jacobi	$G_J = (C^T Q_D C)^{-1}\, C^T F C$
Scaled Laplacian	$G_d = (dC^T C)^{-1}\, C^T(dI - Q)C$
Transformed SOR	$G_\omega = [C^T(Q_D - \omega Q_L)C]^{-1}\, C^T[\omega Q_U + (1-\omega)Q_D]C$

Just as with the standard Jacobi method, this leads to the iteration:

$$\begin{aligned} C^{\mathrm{T}} Q_{\mathrm{D}} C \mathbf{x}^{n+1} &= C^{\mathrm{T}}(Q_{\mathrm{L}} + Q_{\mathrm{U}})\, C\mathbf{x}^{n} + C^{\mathrm{T}}\mathbf{b} \\ &= C^{\mathrm{T}} F C \mathbf{x}^{n} + C^{\mathrm{T}}\mathbf{b}. \end{aligned} \tag{21}$$

Since C has full rank and Q_{D} is a diagonal matrix of positive entries, $(C^{\mathrm{T}} Q_{\mathrm{D}} C)$ is positive definite symmetric and hence invertible. This is the *transformed Jacobi method.*

2.2.3. *The Scaled Laplacian Methods*

These are obtained in exactly the same way as the transformed Jacobi methods except that instead of using $Q = Q_{\mathrm{D}} - F$, the splitting $Q = dI - (dI - Q)$ is used. Hence we have

$$dC^{\mathrm{T}} C \mathbf{x}^{n+1} = C^{\mathrm{T}}(dI - Q)\, C\mathbf{x}^{n} + C^{\mathrm{T}}\mathbf{b}. \tag{22}$$

This is a regular splitting of Q only when $(dI - Q)$ is a non-negative matrix. For $d = 1$, the *scaled Laplacian method* becomes the transformation of Richardson's RF method. If the diagonal matrix Q_{D} is a scalar matrix, sI, the scaled Laplacian method with $d = s$ and the transformed Jacobi method are identical. The method takes its name from the fact that $dC^{\mathrm{T}}C$ is a scalar multiple of the discrete Laplacian matrix, the matrix obtained by using second-order centered differences to approximate the Laplacian.

Note that the sequence of inner systems (16) produced by the scaled Laplacian method (22) is, in essence, a sequence of discrete Laplace systems whose solutions will be proven to converge to the solution of the Navier–Stokes dual variable system.

2.2.4. *The Transformed SOR Method*

We first scale the dual variable equation by ω and then split Q as in (17) to obtain

$$-\omega C^{\mathrm{T}} Q_{\mathrm{L}} C\mathbf{x} = \omega C^{\mathrm{T}} Q_{\mathrm{U}} C\mathbf{x} - \omega C^{\mathrm{T}} Q_{\mathrm{D}} C\mathbf{x} + \omega C^{\mathrm{T}}\mathbf{b}.$$

Adding $C^{\mathrm{T}} Q_{\mathrm{D}} C\mathbf{x}$ to each side yields

$$C^{\mathrm{T}}(Q_{\mathrm{D}} - \omega Q_{\mathrm{L}})\, C\mathbf{x} = C^{\mathrm{T}}[\omega Q_{\mathrm{U}} + (1-\omega) Q_{\mathrm{D}}]\, C\mathbf{x} + \omega C^{\mathrm{T}}\mathbf{b}. \tag{23}$$

It is shown in [26, 27] that the matrix $C^{\mathrm{T}}(Q_{\mathrm{D}} - \omega Q_{\mathrm{L}})C$ is nonsingular for any choice of $\omega \in [0, 1]$. The nonsingularity of $C^{\mathrm{T}}(Q_{\mathrm{D}} - \omega Q_{\mathrm{L}})C$ can be extended to ω in a neighborhood of 1 by a continuity argument.

The methods in Table I were tested [27] on a Poiseuille flow problem along with point Jacobi, point Gauss–Seidel, and point SOR and several other methods. Among the methods not listed in Table I, only SOR and Gauss–Seidel converged for the entire range of test parameters which included various time steps and

Reynold's numbers. The unsuccessful methods are not presented here; see [27]. Also, the point methods were found to be inferior for this simple problem for the following reasons. First, for the 13-point stencil, the point Jacobi and the point SOR methods require the unknowns to be reordered according a three-color scheme in order to vectorize. Because of this, the inner loops are shortened which reduces the vectorization benefit for problems with less than 192 dual variables on a side of the region. Second, they used significantly more iterations than the other convergent methods. The relative merits of the new methods are discussed below.

The stress-convection method requires the solution of a system whose coefficient matrix has 13 non-zero entries per row and is PDS in the special case of uniform mesh-spacing. The matrix $C^{T}QC$ itself has only 13 non-zero entries per row, but is not symmetric. If the system (18) is PDS, then special techniques can be used for its solution. Note that $C^{T}(Q_{D}-F_{S})C=-C^{T}F_{S}C+C^{T}DC$, where D is a diagonal matrix of positive entries, and $-C^{T}F_{S}C$ is a positive multiple of the discrete biharmonic operator. In general, the coefficient matrix is not PDS and it requires as much work to solve as the dual variable system (12). Hence it appears that this iteration is useless in the general case. The storage for $C^{T}(Q_{D}-F_{S})C$ is much larger than for any of the coefficient matrices of the other methods which have at most five non-zero entries per row. Furthermore, one must store both F_{S} and F_{c} instead of just the matrix F. Finally, as the viscosity approaches zero, the stress–convection iteration matrix reduces to the transformed Jacobi iteration matrix and the two methods exhibit similar convergence rates. However, the former is more costly.

convergence criteria and relative to the asymptotic convergence rate. This is not surprising since the analytic solution of the Poisueille flow has no convective effects.

Another method that is hard to implement is the transformed SOR method. Unlike the related methods, SOR and block SOR, the matrix on the left side of the iteration (23) is not block lower triangular. In fact, this matrix has non-zero super-diagonals. There is no simple method to solve a system with this coefficient matrix except Gaussian elimination. While the amount of work required to solve this system is less than solving the dual variable system itself, it requires the same order of magnitude of arithmetic operations. Furthermore, both the transformed Jacobi and scaled Laplacian methods can be accelerated to obtain convergence rates that are comparable to the best that this method can be expected to produce. See Section 2.4.

The transformed Jacobi was the second best among methods tried on the Poiseuille flow problem relative to the number of iterations required to meet a certain convergence criterion and relative to the asymptotic convergence rate. Additionally, there are a variety of means available to accelerate the convergence of this basic method which are discussed in Section 2.4. This method requires the solution of a discrete Poisson equation, which can be handled very efficiently [9, 16, 30]. The matrix, $P=C^{T}Q_{D}C$, with five non-zero diagonals must be stored;

this is more than for the point methods, but not a great deal more if the solution of the inner system is carried out by iterative means. Finally, the method vectorizes extremely well and parallelizes if the solver of the inner system

$$P\mathbf{x}^{k+1} = \mathbf{r} \tag{24}$$

parallelizes. The matrix is also a positive definite symmetric M-matrix and so all methods based on regular splittings, such as point and block Jacobi and SOR, are convergent for (24). ADI is also well suited to solve the system, since the system is block tridiagonal with main diagonal blocks which themselves are tridiagonal. The inner system (24) also yields to solution by reduced system conjugate gradient, RSCG, which is available in the highly reliable commercial package ITPACK.

The scaled Laplacian methods were not tested in the Poiseuille flow study, but are very similar to the transformed Jacobi method, both in performance, storage, vectorization and acceleration possibilities. The inner system (24) of the scaled Laplacian method has $P = dC^{\mathrm{T}}C$. It was already pointed out that $dC^{\mathrm{T}}C$ is a scalar multiple of the discretized form of the Laplacian obtained by using second centered differences and a uniform mesh. In this case, (24) is the discrete Poisson equation. There are a variety of methods available for the solution of this system in addition to those methods for the inner system of the transformed Jacobi [9]. As with the transformed Jacobi, these methods vectorize and parallelize naturally.

Among all the methods reviewed above, the ones with the fewest disadvantages are the transformed Jacobi and the scaled Laplacian.

2.3. *Convergence Results*

We present two results about the convergence of the transformed Jacobi and the scaled Laplacian methods. The first result applies to both methods, and the latter applies only to the scaled Laplacian method.

THEOREM 2.3.1. *If Q and Q^{T} are diagonally dominant, and either is strictly diagonally dominant, then the transformed Jacobi and scaled Laplacian methods are convergent.*

Proof. See [26, 27].

The next two results give much less restrictive conditions for the convergence of the scaled Laplacian method. First we show that the scaled Laplacian method can be used to solve the dual variable system for all problems which can be solved by any variant of conjugate gradient.

THEOREM 2.3.2. *If Q is PDR, then the scaled-Laplacian iteration matrix, G_d, is convergent for d sufficiently large.*

Proof. See [27].

Next, we note that the scaled Laplacian method can be used to solved an even larger class of problems than those characterized by the condition that Q is PDR.

THEOREM 2.3.3. *For uniform mesh spacing, the scaled-Laplacian method, with* $d \geqslant \|Q_D\|_\infty$, *is convergent to the solution of the dual variable system whenever* $(Q + [d/(d-1)] \cdot I)^T$ *is strictly diagonally dominant.*

Proof. See [26, 27].

Note that Q need not be PDR to satisfy the hypotheses of Theorem 2.3.3. For example, if Q is given by

$$Q = \begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}, \quad \text{then} \quad (1 \quad 1)\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ 1 \end{bmatrix} = -1.$$

Thus, Q is not PDR. However, $(Q + [d/(d-1)] \cdot I)^T$ is strictly diagonally dominant for any $d > 1$. Also, note that the hypotheses of Theorem 2.3.3 are easier to verify than those of Theorem 2.3.2; the result is of greater practical utility than Theorems 2.3.2 and 2.1.1.

Under these hypotheses. $Q = I + \Delta t M$, where M is given in Appendix A; thus $d \geqslant (1 + \Delta t \cdot m_{ii}) > 1$ for all i. Therefore, $d/(d-1) > 1$ always because the denominator is always positive. A comparison of Theorem 2.1.2 (or 2.3.1) with Theorem 2.3.3 shows that the bound on the time step given by the latter is more than twice the bound required by the former for guaranteeing invertibility of $C^T Q C$ in the uniform mesh spacing case. Indeed, if $\Delta t_0 := \sup\{\Delta t \mid (I + \Delta t M)^T$ is diagonally dominant$\}$, then $\sup\{\Delta t \mid (I + \Delta t M)^T + [d/(d-1)] \cdot I$ is diagonally dominant$\} = \Delta t_0(2d-1)/(d-1) > 2\Delta t_0$. Notice that when d is only slightly larger than one, $(2d-1)/(d-1) = 2 + 1/(d-1)$ can be much larger than 2.

Q is an M-matrix and $(dI - Q) \geqslant (Q_D - Q) \geqslant 0$ for $d \geqslant \|Q_D\|_\infty$. Therefore, $d^{-1}(dI - Q)$ has a larger spectral radius than that of $Q_D^{-1}(Q_D - Q)$ [30, pp. 90–91]; hence its asymptotic convergence rate is slower. Since these are the untransformed versions of the scaled Laplacian and transformed Jacobi, one would expect the convergence rate of the transformed Jacobi to be higher than that of the scaled Laplacian. In spite of the fact that it has not yet been proven, numerical experiments have always borne this out. This is the main reason that the transformed Jacobi was implemented in the DUALIT computer code instead of the scaled Laplacian.

2.4. *Acceleration Techniques*

The convergence rate of the basic methods discussed in Section 2.2 can be greatly enhanced by applying acceleration methods. Though many techniques are available, [17, Chap. 3], we consider only the polynomial acceleration technique known as second-order Richardson. This technique has a dramatic effect on the converge of the transformed Jacobi method. The dual variable system is denoted

$$B\mathbf{x} = \mathbf{b}, \tag{25}$$

where for the kth time step $B = C^{\mathrm{T}}Q^{k}C$ is the $(L-N)\times(L-N)$ coefficient matrix. We can write $B = S - (S - B)$, where S is a nonsingular matrix called the *splitting matrix*. Suppose that the iterative method for solving (25) is given by

$$\mathbf{x}^{n+1} = G\mathbf{x}^{n} + \mathbf{k}, \tag{26}$$

where $\mathbf{k} = S^{-1}\mathbf{b}$ and $G = S^{-1}(S - B) = I - S^{-1}B$ is the iteration matrix. Using G, one may write down the so-called related system

$$(I - G)\mathbf{x} = \mathbf{k} \tag{27}$$

which has the same solution as (25).

Polynomial acceleration techniques attempt to increase the rate of convergence of an iterative process by creating, from the sequence $\{\mathbf{x}^{n}\}$, a new sequence of iterates, $\{\mathbf{w}^{n}\}$, which approaches the solution faster than the original sequence. The new sequence, $\{\mathbf{w}^{n}\}$, is defined by

$$\mathbf{w}^{n} = \sum_{k=0}^{n} a_{n,k}\mathbf{x}^{n}, \qquad \text{for } n \geqslant 0, \quad \text{where } \sum_{k=0}^{n} a_{n,k} = 1, \text{ for } n \geqslant 0. \tag{28}$$

The error vectors are given by

$$\mathbf{e}_{w}^{n} = \mathbf{w}^{n} - \mathbf{x} \qquad \text{and} \qquad \mathbf{e}_{x}^{n} = \mathbf{x}^{n} - \mathbf{x}, \tag{29}$$

where $\mathbf{x}$ is the solution of (25). From (26) and (27), $\mathbf{e}_{x}^{n+1} = \mathbf{x}^{n+1} - \mathbf{x} = G(\mathbf{x}^{n} - \mathbf{x}) = \cdots = G^{n+1}(\mathbf{x}^{0} - \mathbf{x}) = G^{n+1}\mathbf{e}_{x}^{0}$; then from (28),

$$\mathbf{e}_{w}^{n} = P_{n}(G)\mathbf{e}_{w}^{0},$$

where $P_{n}(G) = a_{n,0}I + a_{n,1}G + \cdots + a_{n,n}G^{n}$.

Since polynomial acceleration in this form uses large amounts of both storage and computations, simpler forms, especially three-term recursions, are used whenever possible. As shown in Hageman and Young [17], if $\{P_{n}\}$ satisfies

$$\begin{aligned} &P_{0}(x) = 1, \\ &P_{1}(x) = c_{1}x - c_{1} + 1, \\ &P_{n+1}(x) = r_{n+1}(c_{n+1}x + 1 - c_{n+1})\,P_{n}(x) + (1 - r_{n+1})\,P_{n-1}(x), \end{aligned}$$

then the polynomial acceleration method (28) may be written as

$$\begin{aligned} &\mathbf{w}^{1} = c_{1}(G\mathbf{w}^{0} + \mathbf{k}) + (1 - c_{1})\mathbf{w}^{0}, \\ &\mathbf{w}^{n+1} = r_{n+1}[c_{n+1}(G\mathbf{w}^{n} + \mathbf{k}) + (1 - c_{n+1})\mathbf{w}^{n}] + (1 - r_{n+1})\mathbf{w}^{n-1} \end{aligned} \tag{30}$$

for $n = 1, 2, 3, \ldots$. Many methods have such three-term recursions; most notable are the SOR, second-order Richardson, Chebyschev, and conjugate-gradient (CG) methods.

The second-order Richardson extrapolation is a polynomial acceleration method obtained by setting $r_n = \omega$ and $c_n = 1$ for $n \geqslant 0$. This yields

$$\mathbf{w}^{n+1} = \omega[G\mathbf{w}^n + \mathbf{k}] + (1-\omega)\mathbf{w}^{n-1}. \tag{31}$$

It was shown by Golub and Varga [13], that (31) is identical to SOR applied to the system:

$$\begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} = \begin{bmatrix} 0 & G \\ G & 0 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{z} \end{bmatrix} + \begin{bmatrix} \mathbf{k} \\ \mathbf{k} \end{bmatrix}. \tag{37}$$

When B is PDS, the optimal value for ω is known to be

$$\omega_b = \frac{2}{1 + \sqrt{1 - s^2(G)}}.$$

The Richardson extrapolation method (31) requires only one extra vector of storage over the basic method and very little extra computation. The convergence rate obtained is of SOR-type but it vectorizes without the need to reorder the variables as is required for SOR in many applications. If G were symmetric, the coefficient matrix of system (32) would be symmetric. By Varga [30, Theorem 4.4], the spectral radius of the SOR iteration matrix for (32) would be $(\omega_b - 1)$. Thus the asymptotic convergence rate for accelerated method (31) is $R_\infty = -\ln(\omega_b - 1)$. Since $(\omega_b - 1)$ is strictly decreasing with decreasing $S(G)$, the best choice of G is the one with the smallest spectral radius. From the discussion in Section 2.3, we expect G_J to have smaller spectral radius than G_d. When the eigenvalues of G^2 are not real and non-negative, little is known about the convergence rate of SOR applied to (32).

The Richardson extrapolation scheme for both the scaled Laplacian and transformed Jacobi can be derived directly from the corresponding regular splittings for Q. For a discretely divergence free sequence $\{\mathbf{w}^n\}$, $\mathbf{w}^n = C\gamma^n$ for a unique $\gamma^n \in R^{L-N}$. For the transformed Jacobi method, we begin with the Jacobi iteration arising from the associated regular splitting (17) for Q:

$$\mathbf{v}^{n+1} = J\mathbf{v}^n + \mathbf{b}^*, \qquad J = Q_{\mathrm{D}}^{-1} F.$$

Apply second-order Richardson extrapolation to accelerate:

$$\mathbf{w}^{n+1} = \omega[J\mathbf{w}^n + \mathbf{k}] + (1-\omega)\mathbf{w}^{n-1}.$$

This can be rewritten as

$$Q_{\mathrm{D}}\mathbf{w}^{n+1} = \omega[F\mathbf{w}^n + Q_{\mathrm{D}}\mathbf{k}] + (1-\omega)\, Q_{\mathrm{D}}\mathbf{w}^{n-1}.$$

Take the discrete curl by multiplying by C^{T} to obtain

$$C^{\mathrm{T}} Q_{\mathrm{D}}\mathbf{w}^{n+1} = \omega[C^{\mathrm{T}} F\mathbf{w}^n + k^*] + (1-\omega)\, C^{\mathrm{T}} Q_{\mathrm{D}}\mathbf{w}^{n-1},$$

where $\mathbf{k}^* = C^T Q_D \mathbf{k}$. Replace mass velocities by dual variables to obtain

$$C^T Q_D C \gamma^{n+1} = \omega[C^T F C \gamma^n + \mathbf{k}^*] + (1-\omega) C^T Q_D C \gamma^{n-1}.$$

Multiply by $(C^T Q_D C)^{-1}$:

$$\gamma^{n+1} = \omega[G_J \gamma^n + \mathbf{k}] + (1-\omega)\gamma^{n-1}.$$

This shows that *the Richardson accelerated transformed Jacobi is just the dual variable transformation of the Richardson accelerated Jacobi.* The derivation for the scaled Laplacian is similar.

3. Numerical Results

The computer code DUALIT is a highly optimized and vectorized program which solves the incompressible Navier–Stokes problems described in Section 1.1. On a given time step, once the particular solution has been found and the coefficient matrix of the momentum equation formed, the solver carries out the transformed Jacobi iteration to find the vector of dual variables. The solver uses second-order Richardson extrapolation to accelerate the convergence of the transformed Jacobi iteration and solves the inner system (24) with reduced conjugate gradient, RSCG. After the dual variable system has been solved, the mass velocities are recovered and the next time step started.

In these dual variable codes, it is important to measure how well the dual variable system is solved. If it were solved exactly, the vector of discrete pressure drops across links, $\mathbf{d}$, would actually be $A^T\mathbf{p}$, and we would have $C^T\mathbf{d} = C^T A^T \mathbf{p} = \mathbf{0}$ by definition. We use the maximum of the sum of pressure drops around the boundaries of the countries, $\|C^T\mathbf{d}\|_\infty$, as a measure of how well the system is solved.

In Sections 3.1 and 3.2, two test cases for DUALIT are presented. The test problems were run on CRAY/XMP computers at both the Pittsburgh Supercomputer Center and the Idaho National Engineering Laboratory. Some conclusions are drawn about the effectiveness of the DUALIT algorithm.

3.1. *The Driven Cavity*

This problem has become a standard problem in the literature and numerical solutions are given in [7, 12]. The flow region is the unit square illustrated in Fig. 4. The problem specifications are summarized as follows:

Initial conditions:

$$u(x, y, 0) = 0.0 \quad \text{for} \quad 0 < x < 1, 0 < y < 1$$
$$v(x, y, 0) = 0.0 \quad \text{for} \quad 0 < x < 1, 0 < y < 1.$$

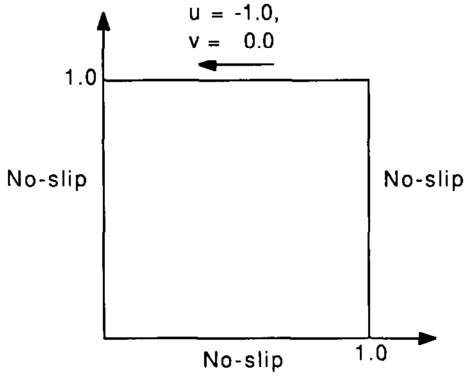

FIG. 4. Driven cavity.

Boundary conditions: no-slip walls: $x=0$, $x=1$, $0<y<1$; and $y=0$, $0<x<1$; for $t>0$.

Specified velocity:

$$u(x, 1, t) = -1.0 \quad \text{for} \quad 0 \leqslant x \leqslant 1, t \geqslant 0$$
$$v(x, 1, t) = 0.0 \quad \text{for} \quad 0 \leqslant x \leqslant 1, t \geqslant 0.$$

Fluid properties: density, $\rho = 1.0$; viscosity, $\mu = 0.025$.

With these specifications, the Reynold's number is 40 and the equations reduce to

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

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = 0.025\Delta u - \frac{\partial p}{\partial x}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = 0.025\Delta v - \frac{\partial p}{\partial y}.$$

TABLE II

Driven Cavity Problem

Size $N \times N$	Outer iterations	Inner iterations	Time in seconds
10×10	72	338	0.39
20×20	135	1358	1.61
30×30	201	2879	4.04
40×40	205	4843	7.72
50×50	254	5579	12.65

TABLE III

Driven Cavity Data from CRAY/XMP48

30 × 30 Cavity, optimal $\omega = 1.38$

Step	1	2	3	4	5	6	7	8	9	10
$10^5 \cdot$ Drop	2.49	1.87	1.74	16.6	17.4	5.85	4.79	7.31	5.87	4.70
Outer	89	59	39	2	2	2	2	2	2	2
Inner	1348	924	474	0	0	43	42	2	44	2
Time	0.940	0.687	0.498	0.264	0.264	0.279	0.278	0.265	0.278	0.278

Total time: 4.039s

40 × 40 Cavity, optimal $\omega = 1.41$

Step	1	2	3	4	5	6	7	8	9	10
$10^5 \cdot$ Drop	4.42	3.85	304	288	239	2.38	7.02	7.53	2.36	1.81
Outer	119	70	2	2	2	2	2	2	2	2
Inner	2491	1346	1	2	3	879	0	0	61	60
Time	2.242	1.484	0.476	0.471	1.12	0.467	0.469	0.469	0.501	0.499

Total time: 7.72s

For the test runs, the region was overlaid with 10 × 10, 20 × 20, 30 × 30, 40 × 40, and 50 × 50 grids with a uniform mesh gauge. A time step of 20.0 s was chosen for all problems. The tolerance for the convergence of the inner iteration was set at 10^{-6} (both the absolute and relative error tolerances). RSCG was used to solve the inner system. The solutions produced by DUALIT and DUVAL [10] agreed to four significant places.

The results of these test runs are summarized in Table II. *Outer* is the number of outer iterations taken by the accelerated transformed Jacobi method to attain the convergence condition. *Inner* is the cumulative number of inner iterations used to solve the inner system on all the transformed Jacobi iterations. *Time* is the number

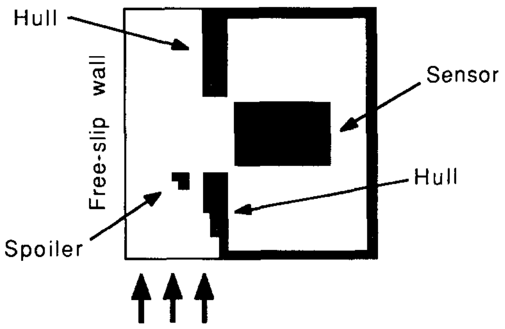

FIG. 5. The aircraft cavity.

of seconds required to complete the computations of 10 time steps including input, initialization, and computations done before the time loop. More detailed information is given in Table III for the cases of $N=30$ and $N=40$.

3.2. *Aircraft Cavity*

In [12], a simulation of the flow of air along the exterior of an aircraft and into an opening in the fuselage is analyzed numerically. Inside the cavity, there is a sensor which is treated as a blockage. In front of the opening in the fuselage is a ramp or spoiler which causes the jetstream of air to shoot up over the opening and reduces the amount of flow into the cavity (see Fig. 5). The inlet velocities on the bottom of the region correspond to a free stream Mach number of 0.75 while pressures of 14.7 psi are specified at the outlet on the top of region. The walls of the cavity and sensor are no-slip walls. The DUALIT code cannot handle walls on the interior of the region; therefore, the fuselage, spoiler, and sensor are approximated by adding large friction factors f_i (see (4)) to the appropriate momentum equations to force the mass-velocities to be zero.

The solutions achieved by DUALIT and DUVAL agree to two significant places throughout the transient. Velocity magnitude contours are given in Fig. 6 at time 25 s. From the picture, one can see the effectiveness of using friction factors to simulate the interior blockages. Figure 7 shows the streamlines of the flow at 25.0 s. As expected, the spoiler causes most of the air to stream up and over the opening. There is a small vortex just above and to the left of the sensor.

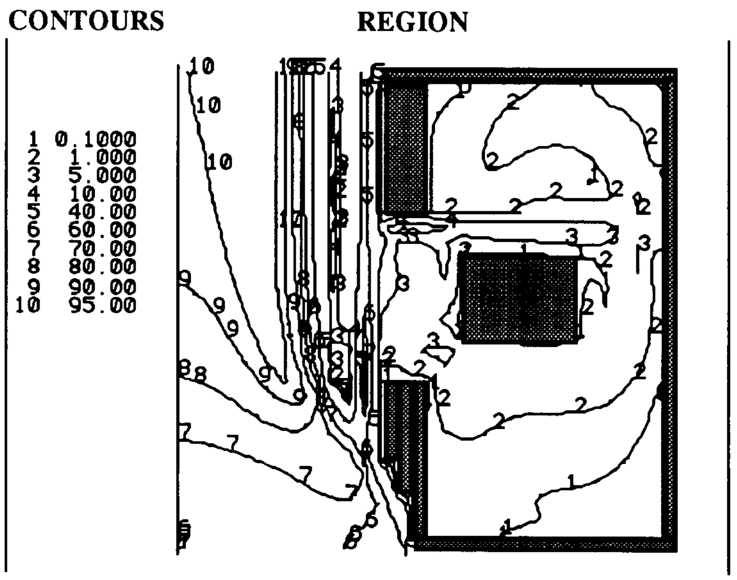

FIG. 6. Aircraft cavity velocity magnitude contours at 25.0 s.

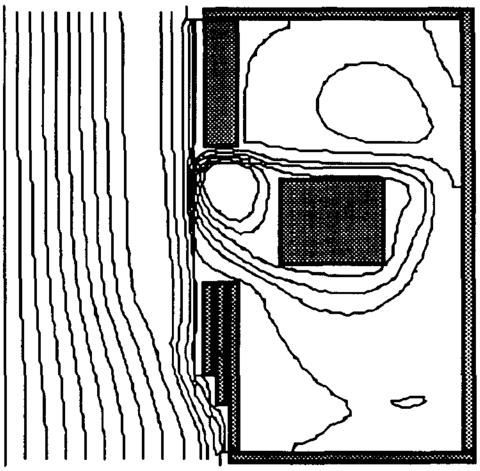

FIG. 7. Aircraft cavity streamlines at 25.0 s.

The data for the driven cavity problem indicate that the time required to complete 10 time steps is approximately a linear function of the number of variables. An experiment to see if this was indeed so was conducted with the more difficult aircraft cavity problem. Three different sized grids were used: 30 × 30, 60 × 60, and 70 × 70. A time step of 0.5 and convergence tolerance of 10^{-5} were used in all three runs. The results are given in Fig. 8 and the data for the three runs are given in

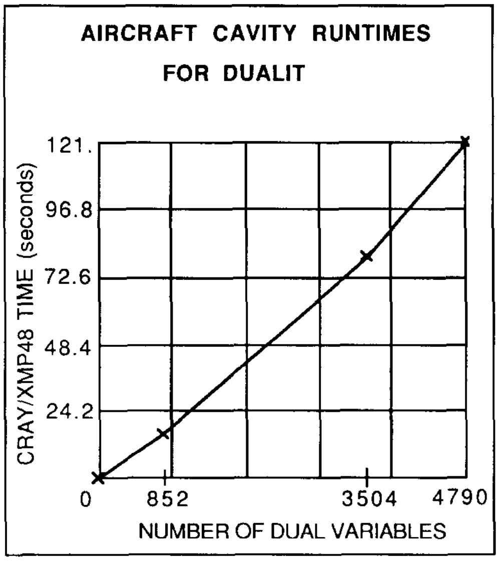

FIG. 8. DUALIT aircraft cavity runtime.

TABLE IV

Aircraft Cavity Data

Size	ω	Outer iterations	Inner iterations	Time
30×30	1.06	2142	26369	16.14
60×60	1.03	2344	54806	80.62
70×70	1.03	2775	59876	121.32

Table IV. The curve shown in Fig. 8 is almost linear. However, no search was made for the optimal extrapolation parameter; so it is possible to improve the runtimes for the 60×60 and 70×70 problems.

3.3. *Comparisons of DUALIT with the Direct Solver DUVAL*

We now compare the performance DUALIT with the direct solver DUVAL, which uses an efficient frontal method to solve the same system of dual variables. Version 23 of DUVAL was described in Ref. [10, 20] and is the version upon which the comparisons are based.

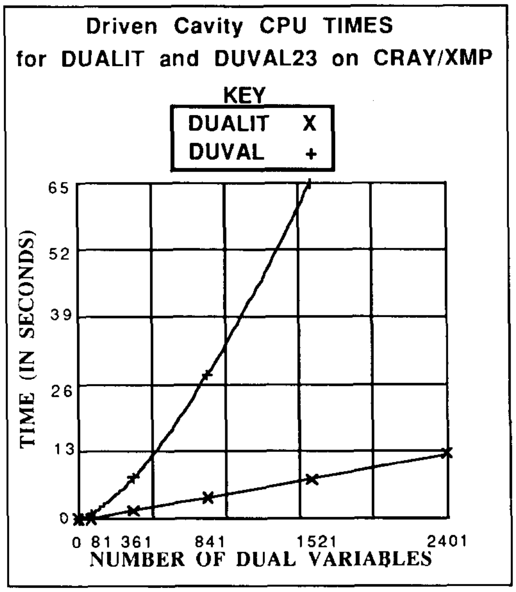

FIG. 9. Comparison of DUALIT and DUVAL CPU times.

TABLE V

Comparisons of DUALIT and DUVAL CPU Times

N	Number of variables	DUALIT CPU time	DUVAL CPU time	Ratio DUVAL/DUALIT	Ratio DUALIT CPU/ number of variables
10	81	0.39	0.93	2. 4	0.0048
20	361	1.61	8.08	5.0	0.0045
30	841	4.07	28.16	6.9	0.0048
40	1521	7.72	86.85	8.5	0.0050
50	2401	12.65	N.A.	N.A.	0.0053

For an $N \times N$ discrete mesh of cells, the formula for the memory requirements of DUALIT is $O(39 \cdot N^2 + 96 \cdot N)$, while the formula for DUVAL is $O(111 \cdot N^2 + 194 \cdot N) + (3N^3$ Disk Storage). Thus DUALIT can solve a $(1.7N) \times (1.7N)$ problem in the same amount of internal memory (not counting peripherals) that DUVAL needs to solve an $N \times N$ problem.

We now compare the execution time of the two codes on two sample problems.

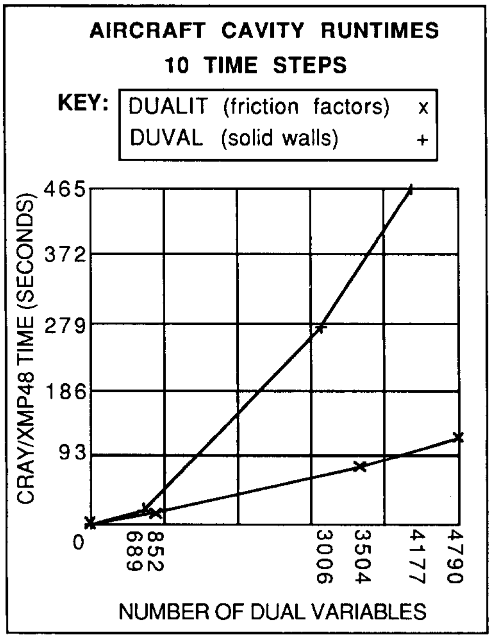

FIG. 10. DUALIT and DUVAL CPU time comparisons for the aircraft cavity problem.

For the driven cavity problem, the convergence tolerances for DUALIT were 10^{-5} for both the inner and outer iterations. With these choices, the velocity fields produced by each code agree to four significant figures for the driven cavity problem. The codes were run for 10 time steps with output at the end of each time step. The comparisons of the execution times for DUVAL and DUALIT are given in Fig. 9. From Fig. 9, we note that DUALIT always uses less computer time than DUVAL to solve the same size driven cavity problem. Moreover, the ratio of DUVAL execution time to DUALIT execution time *increases* with the problem size. The data for Fig. 9 are collected in Table V.

Finally, we compare the CPU times of DUALIT and DUVAL on the aircraft cavity problem (Fig. 10). Three problem sizes were used, 30×30, 60×60, and 70×70; the data is accumulated in Table VI. The problems were run for 50 time steps and output was given at every tenth time step. Both the inner and outer tolerances were set to 10^{-5} for DUALIT and both DUALIT and DUVAL used friction factors to model the blockages. With these settings, the velocity fields produced by the two codes agreed to two decimal places. In neither of the larger problems was the optimal extrapolation factor sought.

3.4. *Conclusions*

Several new iterative linear equation solvers have been developed for the solution of dual variable systems that arise from finite difference discretizations of the 2D incompressible Navier–Stokes problems. They have been shown to be convergent for a larger class of problems than any other known iterative method. The transformed Jacobi method has the optimal asymptotic convergence rate among the class of transformed regular splittings; this is the same rate as obtained with optimal transformed SOR. The new methods vectorize and parallelize in a natural manner. Finally, these new methods require less memory and computation time than the frontal method, which was previously the most efficient method for solving the dual variable system.

TABLE VI

Aircraft Cavity Data

			CRAY/XMP48 time		
$N \times N$	Outer	Inners	DUALIT	DUVAL	Ratio
30×30	2142	26369	16.14	19.40	1.20
60×60	2344	54806	80.62	272.97	3.39
70×70	2775	59876	121.32	464.92	3.83

APPENDIX A: The Finite Difference Equations

The discrete continuity equation is

$$\frac{U_{I,J}^{k+1}-U_{I,J-1}^{k+1}}{\Delta x_I}+\frac{V_{I,J}^{k+1}-V_{I-1,J}^{k+1}}{\Delta y_J}=0. \tag{A.1}$$

Multiplying by $\Delta x_I \Delta y_J$,

$$\Delta y_J(U_{I,J}^{k+1}-U_{I,J-1}^{k+1})+\Delta x_I(V_{I,J}^{k+1}-V_{I-1,J}^{k+1})=0. \tag{A.2}$$

Define $(D_1)_{m,m}=\Delta y_J$ if $w_m=U_{I,J}$ and Δx_I if $w_m=V_{I,J}$; then (A.2) can be written in matrix from as

$$AD_1 w^{k+1}=\mathbf{s}^k, \tag{A.3}$$

where the coefficient matrix A contains entries that are 1 -1, or 0 and is the incidence matrix of an associated graph [2]. The right-hand side, $\mathbf{s}^k$, contains any nonzero boundary information.

The following equation is the discretization of the horizontal momentum equation (2) associated with the flow in the x-direction across the vertical boundary on the right side of the (I, J)th mesh box or flow cell:

$$\begin{aligned}
&\frac{U_{I,J}^{k+1}-U_{I,J}^{k}}{\Delta t}+|U_{I,J}^{k}|\,[H(U_{I,J}^{k})(U_{I,J}^{k+1}-U_{I,J-1}^{k+1})+(1-H(U_{I,J}^{k}))(U_{I,J}^{k+1}-U_{I,J+1}^{k+1})]\\
&\quad+|\bar{V}_{I,J}^{k}|\left[H(\bar{V}_{I,J}^{k})\frac{U_{I,J}^{k+1}-U_{I-1,J}^{k+1}}{(\Delta y_J+\Delta y_{J-1})/2}+(1-H(\bar{V}_{I,J}^{k}))\frac{U_{I,J}^{k+1}-U_{I+1,J}^{k+1}}{(\Delta y_J+\Delta y_{J+1})/2}\right]\\
&\quad\times\mu\left[\frac{U_{I,J+1}^{k+1}}{\Delta x_{I+1}(\Delta x_I+\Delta x_{I+1})/2}-\frac{2U_{I,J}^{k+1}}{\Delta x_I\Delta x_{I+1}}+\frac{U_{I,J-1}^{k+1}}{\Delta x_I(\Delta x_I+\Delta x_{I+1})/2}\right]\\
&\quad\times\mu\left[\frac{8U_{I+1,J}^{k+1}}{(\Delta y_J+\Delta y_{J+1})(\Delta y_{J-1}+2\Delta y_J+\Delta y_{J+1})}\right.\\
&\quad-\frac{8U_{I,J}^{k+1}}{(\Delta y_J+\Delta y_{J+1})(\Delta y_J+\Delta y_{J-1})}\\
&\quad\left.+\frac{8U_{I-1,J}^{k+1}}{(\Delta y_J+\Delta y_{J-1})(\Delta y_{J-1}+2\Delta y_J+\Delta y_{J+1})}\right]\\
&\quad=\frac{P_{I,J}^{k+1}-P_{I,J+1}^{k+1}}{(\Delta x_I+\Delta x_{I+1})/2}+(F_I^{k+1})_{I,J}
\end{aligned} \tag{A.4}$$

where $H(\cdot)$ is the Heaviside operator that is one, if the argument is positive, and zero otherwise.

The vertical momentum equation is similar. The matrix, M^k, contains the coefficients of new time-level velocities for the advection, viscous stress, and drag forces. The coefficient matrix for the pressures turns out to be $D_2 A^{\mathrm{T}}$; see [27], where D_2 is defined by $(D_2)_{m,m} = 2/(\Delta x_I + \Delta x_{I+1})$ if $w_m = U_{I,J}$, and $(D_2)_{m,m} = 2/(\Delta y_J + \Delta y_{J+1})$ if $w_m = V_{I,J}$. In matrix form, the discrete momentum equation is

$$(I + M^k)\mathbf{w}^{k+1} = D_2 A^{\mathrm{T}} \mathbf{p}^{k+1} + \mathbf{b}_2^k. \tag{A.5}$$

Multiplying by D_2^{-1},

$$Q^k \mathbf{z}^{k+1} = A^{\mathrm{T}} \mathbf{p}^{k+1} + \mathbf{b}_1^k, \tag{A.6}$$

where $Q^k = D_2^{-1}((I + M^k) D_1^{-1}$.

References

1. R. Amit, C. G. Cullen, C. A. Hall, G. L. Mesina, and T. A. Porsching, in *Third Int. Conf. on Flow Problems Proc., U. Calgary, June*, 1980.
2. R. Amit, C. A. Hall, and T. A. Porsching, *J. Comput. Phys.* **40**, 183 (1981).
3. Berge and Ghouila-Houri, *Programming, Games and Transportation Networks* (Methuen, London, 1965).
4. M. Berry, M. Heath, I. Kaneko, M. Lawo, R. Plemmons, and R. Word, *Numer. Math.* **47**, 483 (1985).
5. M. Berry and R. Plemmons, in *Proceedings, AMS/SIAM Summer Conference on Linear Algebra in System Theory*, AMS Series on Contemporary Math. (Amer. Math. Soc., Providence, RI, 1985).
6. U. Bulgarelli, G. Graziani, D. Mansutti, and R. Piva, in *Proceedings of the 6th G.A.M.M. Conference, Göttingen, West Germany*, 1985 (GAMM, 1985).
7. J. V. Burkardt, C. A. Hall, and T. A. Porsching, *SIAM J. Alg. Discrete Methods* **7**, 220 (1986).
8. T. Coleman and A. Pothen, *SIAM J. Alg. Discrete Methods* **8**, 544 (1987).
9. F. W. Dorr, *SIAM Rev.* **12**, 248 (1970).
10. R. S. Dougall, C. A. Hall, and T. A. Porsching, Electric Power Research Institute Report EPRI, Report NP-2099, Palo Alto, CA, 1982.
11. H. C. Elman, Dept. of Computer Science, Yale University, Technical Report 266, 1983 (unpublished).
12. A. Frey, C. A. Hall, and T. A. Porsching, *Int. J. Numer. Methods* **24**, 1233 (1987).
13. G. Golub and R. S. Varga, *Numer. Math.* **3**, 147 (1961).
14. J. W. Goodrich and W. Y. Soh, *J. Comput. Phys.* **84**, 207 (1989).
15. D. F. Griffiths, "The Construction of Approximately Divergence-free Finite Elements," *The Mathematics of Finite Elements and Its Applications III*, edited by J. R. Whiteman (Academic Press, London, 1979).
16. K. Gustafson and R. Hartman, *SIAM J. Numer. Anal.* **20**, 697 (1983).
17. L. A. Hageman and D. M. Young, *Applied Iterative Methods* (Academic Press, Orlando, FL, 1981).
18. C. A. Hall, *SIAM J. Alg. Discrete Methods* **6**, 220 (1985).
19. C. A. Hall, J. Peterson, T. A. Porsching, and F. Sledge, *Int. J. Numer. Meth.* **21**, 883 (1985).
20. C. A. Hall, T. A. Porsching, and R. S. Dougall, Electric Power Research Report EPRI Report NP-1416, Palo Alto, CA, 1982.
21. F. Harlow and J. Welch, *Phys. Fluids* **8**, 2182 (1965).
22. M. Heath, R. Plemmons, and R. Ward, *SIAM J. Sci. Statist. Comput.* **5**, 514 (1984).
23. M. R. Hestenes and E. L. Stiefel, *Nat. Bur. Stand. J. Res.* **49**, 409 (1952).
24. B. M. Irons, *Int. J. Numer. Methods* **2**, 5 (1970).

25. D. Mansutti, U. Bulgarelli, R. Piva, and G. Graziani, "A Discrete Vector Potential Method for Unsteady 3-D Navier–Stokes Equations," *10th I.C.N.M.F.M., Beijing*, 1986.
26. S. McCormick (Ed.), *Multigrid Methods* (SIAM, Philadelphia, PA, 1988).
27. G. Mesina, "Iterative Solutions to Navier–Stokes Difference Equations," Ph.D. thesis, University of Pittsburgh and ICMA Technical Report ICMA-88-122, 1988.
28. G. Mesina and C. A. Hall, "Analysis of Iterative Dual Variable Solvers for Navier–Stokes Difference Equations," in preparation.
29. R. Peyret and T. Taylor, *Computational Methods for Fluid Flow* (Springer-Verlag, New York, 1983).
30. T. A. Porsching, J. Murphy, and J. Redfield, *Nucl. Sci. Eng.* **34**, 159 (1982).
31. R. S. Varga, *Matrix Iterative Analysis* (Prentice-Hall, Englewood Cliffs, NJ, 1962).