INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 15, 525–544 (1992)

AN ILU PRECONDITIONER WITH COUPLED NODE FILL-IN FOR ITERATIVE SOLUTION OF THE MIXED FINITE ELEMENT FORMULATION OF THE 2D AND 3D NAVIER–STOKES EQUATIONS

O. DAHL AND S. Ø. WILLE

Department of Informatics, University of Oslo, P.O. Box 1080, Blindern, N-0316 Oslo 3, Norway

SUMMARY

In the present paper, preconditioning of iterative equation solvers for the Navier–Stokes equations is investigated. The Navier–Stokes equations are solved for the mixed finite element formulation. The linear equation solvers used are the orthomin and the Bi-CGSTAB algorithms. The storage structure of the equation matrix is given special attention in order to avoid swapping and thereby increase the speed of the preconditioner. The preconditioners considered are Jacobian, SSOR and incomplete LU preconditioning of the matrix associated with the velocities. A new incomplete LU preconditioning with fill-in for the pressure matrix at locations in the matrix where the corner nodes are coupled is designed. For all preconditioners, inner iterations are investigated for possible improvement of the preconditioning. Numerical experiments are executed both in two and three dimensions.

KEY WORDS Navier–Stokes Mixed formulation Orthomin Bi-CGSTAB Jacobi preconditioning SSOR Incomplete LU Inner iterations

INTRODUCTION

Over the last few years, a great effort have been made to solve large systems of time dependent Navier–Stokes equations in three dimensions. Such efforts have been made in oceanography,[1,2] aerodynamics,[3–5] haemodynamics[6–8] as well as in general fluid dynamics. As progress has been made in designing efficient algorithms[3,9] for generating meshes round arbitrary bodies, attention has been focused on designing large-scale algorithms for solving the Navier–Stokes equations.

For large problems, the use of direct equation solvers is prohibitive due to both the large storage needed and the computational time necessary for solving the problem.[15] Iterative methods have advantages[11–16] compared to direct solvers. However, the success of most iterative equation solvers seems to depend on using a good preconditioner.[17,18]

Several iterative equation solvers for non-symmetric equation systems are available. In this work two equation solvers which have also been used by others[15,17,19] the truncated orthomin method and the Bi-CGSTAB method of Van der Horst,[20–22] have been selected. Different preconditioning methods of iterative equation solvers for flow problems have been subject to extensive studies.[12,14,15,18] Carey *et al.*[15] preconditioned iterative solvers for both penalty and mixed formulation of the Navier–Stokes equations. However, with the mixed formulation they only investigated diagonal preconditioning, and with the penalty formulation they used full

0271–2091/92/170525–20$15.00

Received August 1991
Revised February 1992

factorization of the corresponding Stokes equations as preconditioner. The full factorization was necessary to obtain convergence of the iterative equation solver. The disadvantage with full factorization preconditioning of the Stokes equations is that it requires almost the same memory space as full factorization of the Navier–Stokes equations. Much of the advantage of a reduction in storage requirement and computational work by using an iterative equation solver is thereby lost. Howard *et al.*[23] incorporated 'stabilization' matrices in the equations and investigated incomplete LU factorization. In the present study several preconditioners for the mixed formulation are investigated. The linear equation solvers with different preconditioners have been tested for both the Stokes and the Navier–Stokes equations in two and three dimensions. The numerical experiments have been performed for different Reynolds numbers and different grid resolution. The mixed finite element formulation, first described by Taylor and Hood[24], has been chosen. This mixed element method has certain advantages compared to the penalty method, as it does not introduce 'checkerboard' pressure variations which could well occur with the penalty method.

In the present work, a new incomplete LU preconditioner with slight fill-in is designed specially for preconditioning the Navier–Stokes equations. This preconditioner permits fill-in at certain predefined locations in the pressure coefficient matrix, rather than allowing fill-in when the size of the fill-in coefficient exceeds a certain limit. As the locations of the fill-in are predefined, the housekeeping during the factorization process is considerably reduced. These predefined locations are at the locations in the pressure matrix where the corner nodes are coupled. The corner nodes are coupled if they belong to the same finite element.

THE TEST PROBLEMS

The first test problem concerns the linear Stokes equations. The Stokes equations are given by

$$-\mu\nabla^2\mathbf{v}+\nabla p=0 \quad \text{in } \Omega \tag{1}$$

$$-\nabla\cdot\mathbf{v}=0 \quad \text{in } \Omega \tag{2}$$

where $\mathbf{v}$ is the velocity vector, p is the pressure and μ is viscosity coefficient. In this problem the boundary conditions corresponding to channel flow, Figure 3, are used. The second test problem concerns the Navier–Stokes equations, which are obtained by adding the convective term $\rho\mathbf{v}\cdot\nabla\mathbf{v}$ where ρ is the density, to the Stokes equations,

$$-\mu\nabla^2\mathbf{v}+\rho\mathbf{v}\cdot\nabla\mathbf{v}+\nabla p=0 \quad \text{in } \Omega, \tag{3}$$

$$-\nabla\cdot\mathbf{v}=0 \quad \text{in } \Omega. \tag{4}$$

The boundary conditions for the Navier–Stokes equations correspond to the driven cavity flow given in Figure 3.

In the present mixed finite element formulation, the velocities are approximated by quadratic polynomials and the pressure with linear polynomials. The elements which are used consist of triangles in two dimensions and tetrahedra in three dimensions. For triangles and tetrahedra, the quadratic and linear polynomials are complete. Let the quadratic polynomials be N_i and the linear polynomial L_i. Then by the Galerkin residual method and integration by parts, the finite element formulation of the Stokes equations becomes

$$\int_\Omega \mu\nabla N_i\cdot\nabla\mathbf{v}\,d\Omega+\int_\Omega N_i\nabla p\,d\Omega=0, \tag{5}$$

$$-\int_\Omega L_i\nabla\cdot\mathbf{v}\,d\Omega=0. \tag{6}$$

The minus sign in the continuity equation is important to achieve symmetry of the matrix in the interior of the domain Ω. However, on external boundaries the symmetry is dependent on the boundary conditions. The symmetry is broken if not all velocities are specified at the external boundaries. In many fluid dynamic problems this will be the case as some boundaries have to be open.

In a similar way, the finite element formulation of the Navier–Stokes equations becomes

$$\int_\Omega \mu \nabla N_i \cdot \nabla \mathbf{v}\, d\Omega + \int_\Omega \rho N_i \mathbf{v} \cdot \nabla \mathbf{v}\, d\Omega + \int_\Omega N_i \nabla p\, d\Omega = 0, \tag{7}$$

$$-\int_\Omega L_i \nabla \cdot \mathbf{v}\, d\Omega = 0. \tag{8}$$

Let

$$F_v = \int_\Omega \mu \nabla N_i \cdot \nabla \mathbf{v}\, d\Omega + \int_\Omega \rho N_i \mathbf{v} \cdot \nabla \mathbf{v}\, d\Omega + \int_\Omega N_i \nabla p\, d\Omega, \tag{9}$$

$$F_p = -\int_\Omega L_i \nabla \cdot \mathbf{v}\, d\Omega. \tag{10}$$

By applying Newton's method for solving this non-linear equation system, the set of equations may be written

$$\begin{bmatrix} \dfrac{\partial F_v}{\partial \mathbf{v}} & \dfrac{\partial F_v}{\partial p} \\ \dfrac{\partial F_p}{\partial \mathbf{v}} & 0 \end{bmatrix} \begin{bmatrix} \Delta \mathbf{v}^n \\ \Delta p^n \end{bmatrix} = -\begin{bmatrix} F_v \\ F_p \end{bmatrix}, \tag{11}$$

and

$$\mathbf{v}^{n+1} = \mathbf{v}^n + \Delta \mathbf{v}^n, \tag{12}$$

$$p^{n+1} = p^n + \Delta p^n. \tag{13}$$

Here an initial approximation $(\mathbf{v}^0, p^0)$ must be chosen, and the matrix and right-hand side of equation (11) should be evaluated at the point $(\mathbf{v}^n, p^n)$, in order to get the Newton corrections $(\Delta \mathbf{v}^n, \Delta p^n)$. The iteration process is continued until the solution is considered satisfactory. The stopping criterion which has been used is

$$(\| \Delta \mathbf{v}^n \|^2 + \| \Delta p^n \|^2)^{1/2} \leq \varepsilon (\| \mathbf{v}^{n+1} \|^2 + \| p^{n+1} \|^2)^{1/2}$$

with $\varepsilon = 10^{-4}$ where $\| \cdot \|$ is the Euclidean norm.

STRUCTURE OF THE EQUATION MATRIX

The finite element formulations (5) and (6) of the Stokes problem and the system of equations (11) that must be solved in the Newton process may be written in the form

$$\mathbf{Q}\mathbf{x} = \mathbf{b}, \tag{14}$$

where

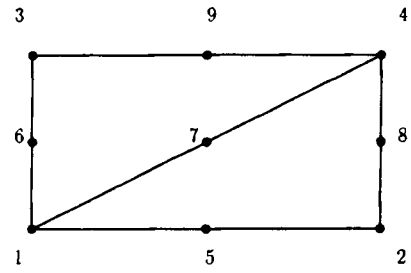
$$\mathbf{Q} = \begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{C} & \mathbf{0} \end{bmatrix}, \qquad \mathbf{x} = \begin{bmatrix} \mathbf{x}_v \\ \mathbf{x}_p \end{bmatrix}, \qquad \mathbf{b} = \begin{bmatrix} \mathbf{b}_v \\ \mathbf{b}_p \end{bmatrix}. \tag{15}$$

For the Stokes problem, $\mathbf{x}_v$ and $\mathbf{x}_p$ contain the node values of the velocity components and pressure, respectively. For the Navier–Stokes problem they contain the Newton corrections of the same quantities. The storage of the matrix $\mathbf{Q}$ is important when an iterative equation solver is used. For large problems only non-zero coefficients should be stored. Several storage schemes exist for storing sparse matrices. If the grid is regular, the different diagonals can be stored as one-dimensional vectors.[16] For irregular grids, a more complex pointing structure is used to identify the rows and coefficients in the matrix used.[25] When solving Navier–Stokes problems for irregular grids, a special storing scheme believed to have several advantages has been developed (Figures 1 and 2). Figure 1 shows the structure of the matrix $\mathbf{Q}$ for a simple two-dimensional grid consisting of two triangular elements. The upper part of the figure shows the grid and numeration of nodes. The corner nodes are numbered first. This way of numbering is obtained by the unstructured grid generation algorithm given by Wille[9] and is advantageous both in storing the matrix and during incomplete LU preconditioning. The indicies above and to the left of the matrix shown in the lower part of Figure 1 refer to the node numbers. The equation matrix has non-zero coefficients at locations where two nodes are coupled in the grid. Two nodes are said to be coupled if they belong to the same finite element. For example, in the grid in Figure 1, nodes 1 and 5 are coupled while nodes 2 and 9 are not. The equation system is symmetric in shape. The matrix $\mathbf{A}$ contains the coefficients associated with the velocity degrees of freedom. The upper and lower parts of the submatrices of $\mathbf{A}$ are stored in separate one dimensional vectors $\mathbf{U}$ and $\mathbf{L}$ as shown in Figure 2. This splitting is advantageous during the preconditioning, as the lower triangular part is accessed by columns and the upper triangular part is accessed by rows during factorization. The pointing structure has two pointing vectors, the first, PAC points to where the node numbers for corresponding rows are stored in the other pointing vector, PAR. Let dim be the spatial dimension for the set of differential equations, then the dimension of each of the submatrices $\mathbf{A}_{ij}$ is [dim × dim]. The position for each of these submatrices in both $\mathbf{U}$ and $\mathbf{L}$ is then easily calculated from the corresponding index in the vector PAR. Corresponding pointing structures are established for the matrices $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{P}$. The matrix $\mathbf{P}$ is initially zero, but fill-in will occur during the ILU preconditioning. The dimension of $\mathbf{B}_{ij}$ is [dim × 1] and $\mathbf{C}_{ij}$ is [1 × dim]. The locations of the submatrices $\mathbf{B}_{ij}$ and $\mathbf{C}_{ij}$ are also easily calculated from the index in the pointing vector PBCR. The fill-in in the submatrices for the pressure $\mathbf{P}_{ij}$ consists of simple scalars. The pointing structure for the $\mathbf{P}$ matrix is equal to the first elements, which correspond to the pointing structure for the corner nodes, of the $\mathbf{B}$ and $\mathbf{C}$ matrices. The pointing structure for the $\mathbf{B}$ and $\mathbf{C}$ matrices can then also be used for addressing the $\mathbf{P}$ matrix. The right-hand side and the solution vector, $\mathbf{b}_v$ and $\mathbf{x}_v$, can also be considered to consist of subvectors $\mathbf{b}_{vi}$ and $\mathbf{b}_{xi}$ with dimension [dim × 1]. These subvectors will represent the right-hand side and solution for node i. The vectors $\mathbf{b}_p$ and $\mathbf{x}_p$ consist of scalars, b_{pi} and b_{pi}, which are the right-hand side of the continuity equation and the pressure in node i. All the submatrices $\mathbf{A}_{ij}$, $\mathbf{B}_{ij}$ and $\mathbf{C}_{ij}$ are stored row by row.

An important point to notice is that for the Navier–Stokes equations all the coefficients in the submatrices $\mathbf{A}_{ij}$ are non-zero. However, for the Stokes problem there are only non-zero coefficients on the diagonal of $\mathbf{A}_{ij}$. This implies that only the diagonals of $\mathbf{A}_{ij}$ need to be stored for the Stokes equations.

PRECONDITIONING

Let $\mathbf{M}$ be a non-singular matrix. The original equation system $\mathbf{Qx}=\mathbf{b}$ can be replaced with $\mathbf{M}^{-1}\mathbf{Qx}=\mathbf{M}^{-1}\mathbf{b}$. The quality of the preconditioner depends very much on the choice of $\mathbf{M}$. The



	1	2	3	4	5	6	7	8	9	1	2	3	4
1	A_{11}	A_{12}	A_{13}	A_{14}	A_{15}	A_{16}	A_{17}	A_{18}	A_{19}	B_{11}	B_{12}	B_{13}	B_{14}
2	A_{21}	A_{22}	0	A_{24}	A_{25}	0	A_{27}	A_{28}	0	B_{21}	B_{22}	0	B_{24}
3	A_{31}	0	A_{33}	A_{34}	0	A_{36}	A_{37}	0	A_{39}	B_{31}	0	B_{33}	B_{34}
4	A_{41}	A_{42}	A_{43}	A_{44}	A_{45}	A_{46}	A_{47}	A_{48}	A_{49}	B_{41}	B_{42}	B_{43}	B_{44}
5	A_{51}	A_{52}	0	A_{54}	A_{55}	0	A_{57}	A_{58}	0	B_{51}	B_{52}	0	B_{54}
6	A_{61}	0	A_{63}	A_{64}	0	A_{66}	A_{67}	0	A_{69}	B_{61}	0	B_{63}	B_{64}
7	A_{71}	A_{72}	A_{73}	A_{74}	A_{75}	A_{76}	A_{77}	A_{78}	A_{79}	B_{71}	B_{72}	B_{73}	B_{74}
8	A_{81}	A_{82}	0	A_{84}	A_{85}	0	A_{87}	A_{88}	0	B_{81}	B_{82}	0	B_{84}
9	A_{91}	0	A_{93}	A_{94}	0	A_{96}	A_{97}	0	A_{99}	B_{91}	0	B_{93}	B_{94}
1	C_{11}	C_{12}	C_{13}	C_{14}	C_{15}	C_{16}	C_{17}	C_{18}	C_{19}	P_{11}	P_{12}	P_{13}	P_{14}
2	C_{21}	C_{22}	0	C_{24}	C_{25}	0	C_{27}	C_{28}	0	P_{21}	P_{22}	0	P_{24}
3	C_{31}	0	C_{33}	C_{34}	0	C_{36}	C_{37}	0	C_{39}	P_{31}	0	P_{33}	P_{34}
4	C_{41}	C_{42}	C_{43}	C_{44}	C_{45}	C_{46}	C_{47}	C_{48}	C_{49}	P_{41}	P_{42}	P_{43}	P_{44}

Figure 1. The upper part of the figure shows a simple two-dimensional grid consisting of two elements. In this grid, the corner nodes are numbered first, then the midedge nodes. The structure of the corresponding equation matrix is shown below. The numbers at the top and to the left of the matrix are nodes numbers. The matrix **P** is initially zero and is used in the ILU preconditioning with fill-in

preconditioning matrix **M** should have the following properties:

(1) **M** is a good approximation to **Q**.
(2) **M** is easily computed.
(3) **M** is reasonably sparse.
(4) Equations of the form $\mathbf{Mx}=\mathbf{c}$ are easily solved.

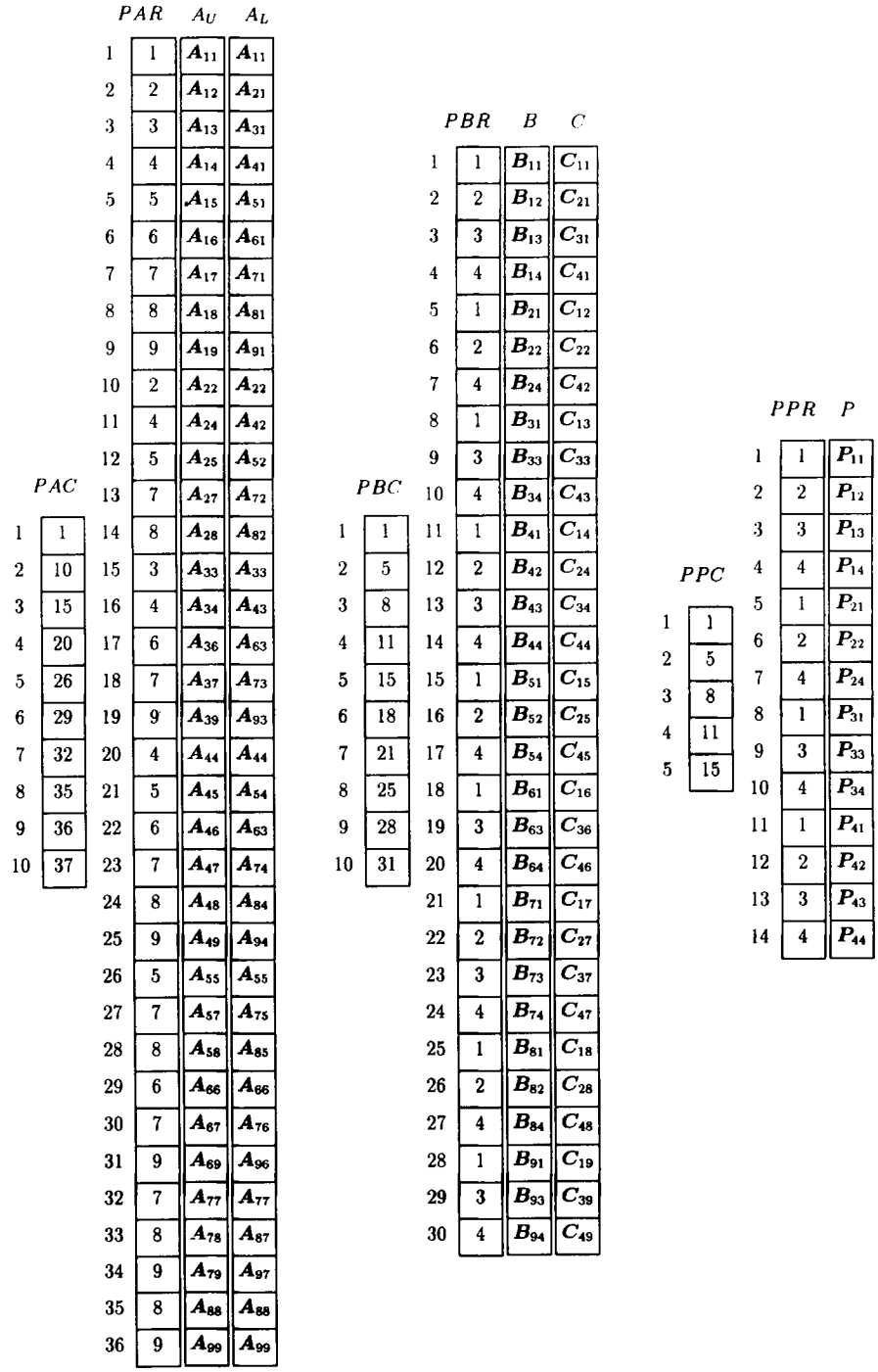


Figure 2. The equation matrix is stored as five one-dimensional vectors. The zero submatrices shown in Figure 1 are not stored. The storage structure requires two pointing vectors, one pointing to where each row begins in the vector which contains the nodes included in that row. The upper part **U** and the lower part **L** of **A** are stored in separate vectors. The same pointing structure can then be used for both **U** and **L**. Similarly, the same pointing structure is used for **B** and **C**. Note that the first of the pointing structure for **B** and **C** can also be used for **P**

There are several ways of selecting **M**. Let $\mathbf{L}_v$, $\mathbf{D}_v$ and $\mathbf{U}_v$ be the lower part, the diagonal and the upper part of **A**. The following preconditioning methods have been considered.

Diagonal preconditioning

$$\mathbf{M}=\begin{bmatrix}\mathbf{D}_v & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}. \tag{16}$$

This corresponds to Jacobian or diagonal preconditioning of **A**, and is referred to as diagonal (preconditioning) in the tables.

SSOR preconditioning

$$\mathbf{M}=\begin{bmatrix}\dfrac{1}{\omega(2-\omega)}(\mathbf{D}_v+\omega\mathbf{L}_v)\mathbf{D}_v^{-1}(\mathbf{D}_v+\omega\mathbf{U}_v) & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}, \tag{17}$$

where the parameter ω is between 1·0 and 2·0. When **A** is symmetric this corresponds to the symmetric successive over-relaxation, SSOR, preconditioning of **A**. The preconditioning is performed by solving a lower and an upper triangular equation system. The pointing structure, which permits the addressing of $\mathbf{U}_v$ by row and $\mathbf{L}_v$ by column, simplifies the solution procedure of the lower and upper triangular equation systems. Let $\mathbf{Mv}=\mathbf{u}$, then

$$\text{Solve } (\mathbf{D}+\omega\mathbf{L})\mathbf{y}=\mathbf{u}, \quad \mathbf{z}=\mathbf{Dy}, \qquad \text{Solve } (\mathbf{D}+\omega\mathbf{U})\mathbf{w}=\mathbf{z}.$$

The SSOR preconditioning requires no extra storage as no coefficients are changed during the preconditioning. The efficiency of the SSOR preconditioning depends on the choice of ω. It has not been tried to find an optimal choice of ω. $\omega=1{\cdot}2$ have been used to produce the results of Table II.

Incomplete factorization of **A**

A can be split as $\tilde{\mathbf{L}}_v\tilde{\mathbf{U}}_v$, where $\tilde{\mathbf{L}}_v\tilde{\mathbf{U}}_v$ is an approximate **LU** factorization of **A**. The precondition matrix **M** for this preconditioner, ILU_v, is

$$\mathbf{M}=\begin{bmatrix}\tilde{\mathbf{L}}_v\tilde{\mathbf{U}}_v & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}. \tag{18}$$

Incomplete factorization of the symmetric part of **A**

For symmetric preconditioning, obviously $\mathbf{L}_v^s=\mathbf{U}_v^s$. The symmetric version ILU_v^s of this preconditioner is then

$$\mathbf{M}=\begin{bmatrix}\tilde{\mathbf{L}}_v^s\tilde{\mathbf{U}}_v^s & \mathbf{0}\\ \mathbf{0} & \mathbf{I}\end{bmatrix}, \tag{19}$$

where $\tilde{\mathbf{L}}_v^s\tilde{\mathbf{U}}_v^s$ is the incomplete factorization of the symmetric part of **A**.

Incomplete factorization of **Q**

Incomplete LU factorization on the complete matrix **Q** (see Figure 1) can be performed if certain fills are accepted. In the present work these fill-ins are associated with the pressure and correspond to the matrix **P** given in Figure 1. Using the previous definition of coupled nodes, the submatrices $\mathbf{A}_{ij}$ are updated during the forward elimination of the matrix **A**, if the two nodes i, j

are coupled in the element graph. The same philosophy is applied to the fill-ins in the pressure matrix $\mathbf{P}$. Let nodes i and j be corner nodes, then the fill-in $\mathbf{P}_{ij}$ are accepted if the nodes i, j are coupled. The general algorithm for forward eliminations and backward substitution is found in the appendix. In this algorithm an inversion of the submatrices $\mathbf{A}_{kk}$ on the diagonal is included. As mentioned earlier, when the Stokes equations are considered, only the diagonal of these submatrices need to be stored. When using the Stokes equations as preconditioner, fill-in will never occur outside this diagonal. The inversion of the diagonal submatrices for the Stokes preconditioner will therefore only consist of inverting the diagonal in these submatrices. The preconditioning matrix of this preconditioner, ILU, then becomes

$$\mathbf{M}=\tilde{\mathbf{L}}\tilde{\mathbf{U}}. \tag{20}$$

Incomplete factorization of the symmetric part of $\mathbf{Q}$

A symmetric version, ILU^s, based on the symmetric part $(\mathbf{Q}+\mathbf{Q}^T)/2$ of $\mathbf{Q}$ of the above preconditioner is

$$\mathbf{M}=\tilde{\mathbf{L}}^s\tilde{\mathbf{U}}^s. \tag{21}$$

In this work, the matrix $\mathbf{Q}$ corresponding to the Stokes problem has been used when computing the preconditioning matrix $\mathbf{M}$. Hence, it has not been necessary to compute a new preconditioning matrix for each Newton step. However, it is surprising that although the matrix $\mathbf{Q}$ is singular, no such singularity was observed in $\mathbf{M}=\tilde{\mathbf{L}}\tilde{\mathbf{U}}$. Here, the lower part of $\mathbf{Q}$, $\tilde{\mathbf{L}}$, has unit diagonal, and it was observed that the diagonal elements of $\tilde{\mathbf{U}}$, corresponding to the zero block matrix in the lower right corner of $\mathbf{Q}$ were all negative, and of the same order of magnitude. The same observations were made for the symmetric factorization $\mathbf{M}=\tilde{\mathbf{L}}^s\tilde{\mathbf{U}}^s$.

INNER ITERATIONS AND EQUATION SOLVERS

The preconditioning can be improved by inner iterations. When the equations are preconditioned by $\mathbf{M}^{-1}$, the equation system to be solved is $\mathbf{M}^{-1}\mathbf{Q}\mathbf{x}=\mathbf{M}^{-1}\mathbf{b}$. During each iteration, the equation $\mathbf{Q}\mathbf{x}=\mathbf{c}$ is solved approximately, while $\mathbf{M}\mathbf{z}=\mathbf{c}$ is solved exactly. A better approximation to the solution of $\mathbf{Q}\mathbf{z}=\mathbf{c}$ can be obtained by iterations on the residual of $\mathbf{r}=\mathbf{c}-\mathbf{Q}\mathbf{z}$. Let the correction to $\mathbf{z}^i$, at each iteration be $\mathbf{s}^i$. Then $\mathbf{z}^i=\mathbf{z}^{i-1}+\mathbf{s}^i$. The new residual vector has the form $\mathbf{r}^i=\mathbf{r}^{i-1}-\mathbf{Q}\mathbf{s}^i$. The correction to the residual vector is the solution of $\mathbf{M}\mathbf{s}^i=\mathbf{r}^{i-1}$. The inner iteration algorithm is given by

$$\begin{aligned}&\mathbf{z}^0=\mathbf{M}^{-1}\mathbf{c}\\&\text{for } i=1 \text{ to } n \text{ do}\\&\quad \mathbf{r}^{i-1}=\mathbf{c}-\mathbf{Q}\mathbf{z}^{i-1},\\&\quad \mathbf{s}^i=\mathbf{M}^{-1}\mathbf{r}^{i-1},\\&\quad \mathbf{z}^i=\mathbf{z}^{i-1}+\mathbf{s}^i.\end{aligned} \tag{22}$$

The algorithm above can be used for improving the convergence rate of all the preconditioning methods described above.[18]

Two linear equation solvers have been tested

(1) The truncated Orthomin method, with 10 search direction vectors. In this paper, this method is referred to as Orthomin (10).[15,19]
(2) The Bi-CGSTAB method of Van der Vorst.[20–22] This method is a new variant of Bi-Conjugate Gradients and has, to some extent, less irregular convergence behaviour than the Conjugate Gradients-Squared (CG-S) method. This method is referred to as CGSTAB in this paper.

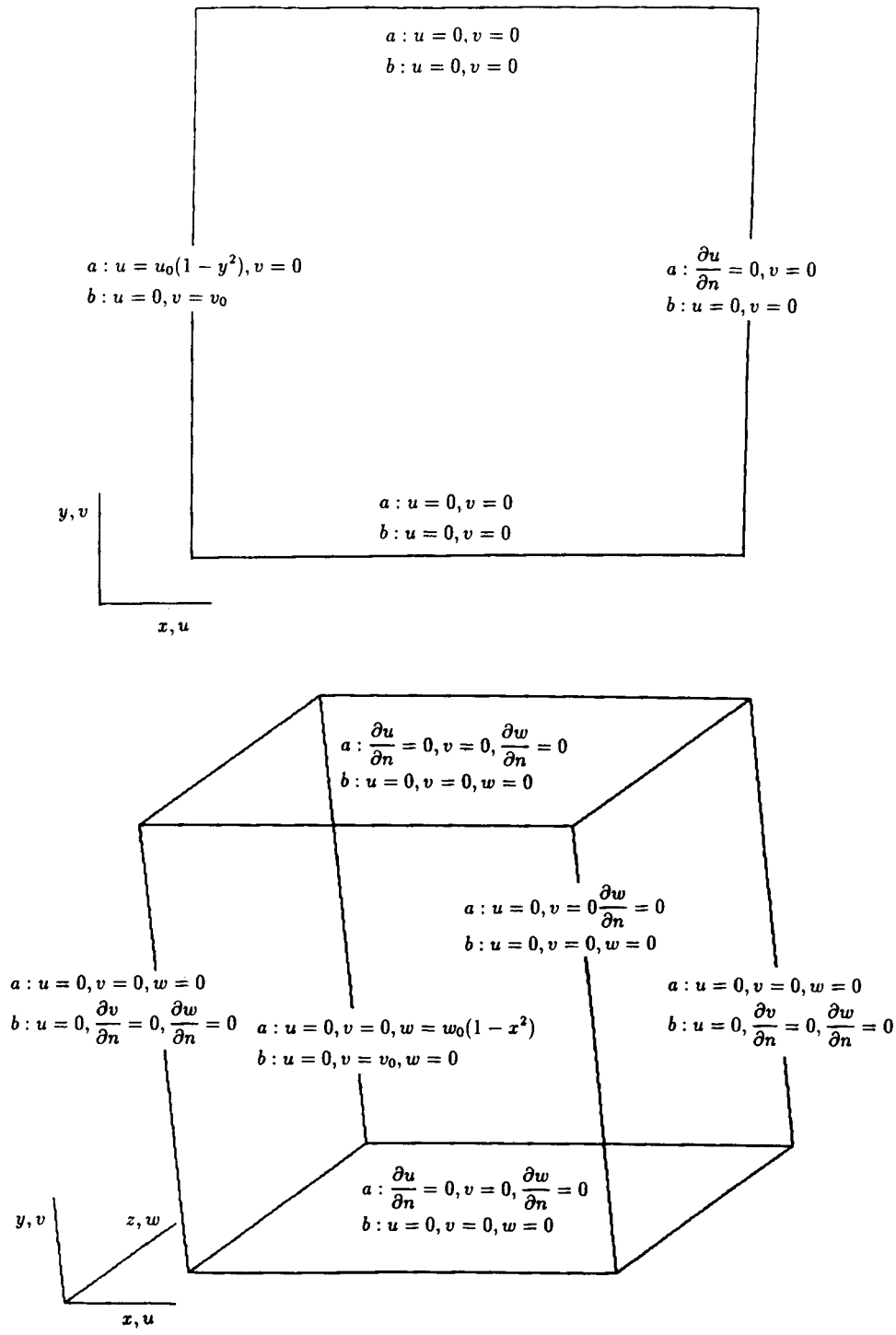


Figure 3. The upper part of the figure shows the boundary conditions for the two-dimensional and the lower part shows the boundary conditions for the three-dimensional test problems. (a) boundary conditions for channel flow for the Stokes equations, (b) the boundary conditions for the non-linear cavity flow problem described by the Navier–Stokes equations

Although the matrix $\mathbf{M}^{-1}\mathbf{Q}$ is singular, the pressure has not been specified at one node, or normalized in any other way, in order to get a unique solution. Iterative solvers do not need such a specification, as this is provided by the initial start vector. In all cases the start vector was the zero vector.

The two linear equation solvers have been tested for several preconditioners and with different grid resolutions for the Stokes equation in two and three dimensions. The following convergence criterion for the linear equation solvers has been used.

$$\frac{\|\mathbf{r}\|}{\|\mathbf{b}\|} < \varepsilon \tag{23}$$

where $\mathbf{r} = \mathbf{b} - \mathbf{A}\tilde{\mathbf{x}}$ is the residual of the computed approximate solution $\tilde{\mathbf{x}}$, of the system $\mathbf{A}\mathbf{x} = \mathbf{b}$. The value of ε was specified to be $\varepsilon = 10^{-4}$.

For the Navier–Stokes equations, only the CGSTAB solver was used. However, since the linear equation solvers sometimes may show a stagnant behaviour, instead of restarting the linear equation solver in the case of the Newton method, the solution $(\mathbf{v}^n, p^n)$ has been updated with corrections $(\mathbf{x}_v^{(i+1)}, \mathbf{x}_p^{(i+1)})$ when

$$(\|\delta\mathbf{x}_v^{(i)}\|^2 + \|\delta\mathbf{x}_p^{(i)}\|^2)^{1/2} < \varepsilon^2(\|\mathbf{v}^n\|^2 + \|p^n\|^2)^{1/2}. \tag{24}$$

Here $\mathbf{x}^{(i)}$ is the ith approximation to the solution of the linear equations, and $\mathbf{x}^{(i+1)} = \mathbf{x}^{(i)} + \delta\mathbf{x}^{(i)}$. This actually happens only in a few cases of the last Newton step, when the ILU^s preconditioning

Table I. The upper table shows initial work in number of multiplications $\times 10^3$, which is the forward elimination of the equation matrix. The lower table shows the work executed in each iteration with Orthomin(10) and CGSTAB for the different preconditioners. The last column shows the amount of iterative work $\times 10^3$ for each inner iteration

Initial work

Grid	ILU$_v$	ILU
5 × 5	40	89
7 × 7	78	173
10 × 10	158	351
15 × 15	355	786
20 × 20	630	1394
50 × 50	3920	8669

Iterative work

	Orthomin(10)			CGSTAB			
Grid	Diagonal	ILU$_v$ SSOR	ILU	Diagonal	ILU$_v$ SSOR	ILU	Inner iterations
5 × 5	17	22	24	18	28	33	15
7 × 7	32	42	47	34	55	63	29
10 × 10	63	83	92	69	109	126	58
15 × 15	139	184	203	153	242	280	129
20 × 20	245	323	357	270	427	495	228
50 × 50	1498	1982	2192	1663	2630	3050	1409

was used. When the non-symmetric ILU preconditioning was used, it happens a few times that, neither one of the stopping criteria was fulfilled, and the iteration was stopped when the number of iterations was larger than 1000.

NUMERICAL RESULTS

The domains with boundary conditions in both the 2D and 3D, for which the algorithms are tested are shown in Figure 3. The domains are divided into triangular or tetrahedral finite elements. The number of elements, nodes and unknowns are given below. Let the division of the regular domain in each direction be n, then

	Grid size	Elements	Nodes	Unknowns
2D	$n \times n$	$2n^2$	$(2n+1)^2$	$2(2n+1)^2+(n+1)^2$
3D	$n \times n \times n$	$6n^3$	$(2n+1)^3$	$3(2n+1)^3+(n+1)^3$

Stokes equations

The viscosity coefficient used in the present experiments was $\mu = 0{\cdot}001$.

The work executed at each iteration for the different preconditioners with Orthomin(10) and CGSTAB are given in Table I. The diagonal and SSOR preconditioners need no initial computations. The ILU type preconditioners need a factorization of the preconditioning matrix

Table II. The number of iterations to achieve convergence with CGSTAB and Orthomin(10) for the Stokes equations. The different preconditioners are diagonal, SSOR, ILU_v on the velocity matrix, ILU with fill-in on the entire matrix and ILU^s on the symmetric part of the entire matrix. The viscosity μ is 10^{-3}. The first number in each column is the number of linear iterations without inner iterations and the second is the number of iterations with one inner iteration

CGSTAB

Grid	Diagonal	SSOR	ILU_v	ILU	ILU^s
5×5	85, 112	78, 73	89, 80	11, 11	10, 13
7×7	152, 185	120, 112	115, 82	17, 14	16, 18
10×10	206, 210	138, 121	128, 109	27, 19	23, 21
15×15	369, 227	209, 189	203, 141	65, 31	37, 25
20×20	619, 307	269, 227	268, 242	135, 57	60, 37

Orthomin(10)

Grid	Diagonal	SSOR	ILU_v	ILU	ILU^s
5×5	495, —	381, 334	338, 259	22, 10	16, 12
7×7	538, —	683, —	—, 299	37, 36	29, 30
10×10	774, 703	—, —	—, —	56, 46	45, 40
15×15	—, 782	—, —	—, —	98, 55	84, 52
20×20	—, 973	—, —	—, —	159, 86	133, 68

before the iterative process is started. However, this factorization is considered as an initialization and this work is not considered as iterative work in solving the linear equations.

In Table II the number of iterations necessary to achieve convergence with Orthomin(10) and CGSTAB for the different preconditioners are listed. In each column, the first number is without inner iterations and the second one is with one inner iteration. Table II shows that the CGSTAB equation solver is far more efficient than Orthomin(10) for all preconditioners. ILU and ILU^s are the best preconditioners in terms of the number of iterations, and also obviously in terms of computational work. Hence, only these were used in the rest of the experiments. One inner iteration also improves the convergence rate. The columns marked with — indicate that the solution is not convergent or the iteration process is stagnant.

In Table III, the Stokes equations are solved with CGSTAB with zero, one, two and three inner iterations. In these experiments only the CGSTAB equation solver and the ILU^s preconditioner were used. The amount of work in each inner iteration is mainly the matrix–vector product $\mathbf{v} = \mathbf{M}^{-1}\mathbf{Qu}$. In each CGSTAB iteration this same product is computed twice. The first number in each column shows the number of iterations and the last number in each column indicates the number of times the above matrix vector product is computed. The results in this table show that one inner iteration is the most efficient.

The results in Table IV show the number of iterations for different three dimensional grids for the Stokes equations. No continuation method was used. The results show that CGSTAB is more efficient than Orthomin (10). They also show that the ILU^s preconditioner is better than the ILU preconditioner.

Table III. The number of CGSTAB iterations with the ILU^s preconditioner for the Stokes equations with zero, one, two and three inner iterations. The first number in each column indicates the number of iterations. The second number indicate total number of matrix vector multiplications $\mathbf{M}^{-1}\mathbf{Ax}$ in CGSTAB, including inner iterations

	Inner iterations			
Grid	Zero	One	Two	Three
10 × 10	22, 44	10, 40	7, 42	6, 48
20 × 20	75, 150	34, 136	25, 150	20, 160
40 × 40	270, 540	106, 424	75, 450	59, 472
80 × 80	699, 1398	304, 1216	207, 1242	169, 1352

Table IV. The number of iterations for Orthomin(10) and CGSTAB with ILU and ILU^s preconditioners for the Stokes equations with different resolutions of the three-dimensional grid. No inner iterations are used

	Orthomin(10)		CGSTAB	
Grid	ILU	ILU^s	ILU	ILU^s
3 × 3 × 3	131	24	26	13
4 × 4 × 4	191	28	37	15
5 × 5 × 5	—	34	68	18
6 × 6 × 6	—	39	92	22
8 × 8 × 8	—	61	140	29
10 × 10 × 10	—	78	253	38

Navier–Stokes equations

The preconditioner was based on only the Stokes equation matrix. In Table V, the number of Newton iterations as well as CGSTAB iterations are given for different grids and Reynolds numbers for unit cavity flow. Clearly, the ILU^s preconditioner is superior to the ILU preconditioner. The solution of the Navier–Stokes equations for Reynolds number 400 and a 10×10 grid is shown as velocity vectors in Figure 4.

The experiments with cavity flow are similar to those done by Carey *et al.*[15] However, Carey *et al.* used the penalty formulation to obtain their results. The penalty formulation differs from the mixed formulation in that the pressure is eliminated from the equation system so the total number of equations is reduced and the equation matrix may become very ill-conditioned. The preconditioner used by Carey *et al.* was a full factorization of the penalty matrix of Stokes equations.

Comparing the results of Carey with those obtained here, the tendency seems to be that fewer Newton iterations are needed with the mixed formulation. Concerning the number of linear iterations a comparison of the results is difficult. Carey reports that in some experiments the Orthomin(5) algorithm was stagnant and failed to converge. This never occurred with the CGSTAB algorithm with the ILU^s preconditioner.

In Table VI the number of Newton and CGSTAB iterations are given for different maximum numbers of iterations in the linear solver at each Newton step. This experiment has also been carried out by Carey *et al.* In the first Newton iteration Carey always got one linear iteration. This is, of course, due to the full factorization of their preconditioner. A comparison with Carey's result shows that both more Newton iteration and more linear iterations are needed with the mixed

Table V. The number of iterations for CGSTAB with ILU and ILU^s preconditioning for Navier–Stokes equations in two dimensions. No inner iterations are used. In the two last columns, the first number in the table is the number of Newton iterations, the second is the total number of CGSTAB iterations. The numbers in brackets are the number of CGSTAB iterations at each Newton step

Grid	*Re*	ILU	ILU^s
5×5	100	4/35 (8, 14, 12, 11)	4/35 (8, 14, 12, 11)
	200	5/83 (11, 19, 19, 18, 16)	5/79 (11, 19, 19, 19, 11)
	300	6/136 (12, 27, 30, 32, 32, 3)	6/138 (12, 27, 30, 33, 33, 3)
	400	6/205 (13, 35, 44, 43, 55, 15)	6/208 (13, 35, 52, 43, 52, 13)
	500	7/* (19, 38, 33, >1000, 71, 71, 5)	7/672 (16, 40, 34, 426, 78, 71, 7)
7×7	100	5/65 (12, 18, 17, 16, 2)	5/65 (12, 18, 17, 16, 2)
	200	5/124 (15, 32, 30, 26, 21)	5/126 (15, 32, 30, 27, 22)
	300	5/204 (21, 47, 44, 43, 49)	5/199 (21, 46, 42, 43, 47)
	400	6/350 (23, 62, 78, 65, 67, 55)	6/252 (23, 62, 81, 66, 67, 53)
	500	7/770 (23, 86, 286, 118, 95, 95, 67)	7/661 (23, 86, 155, 121, 96, 97, 83)
10×10	100	5/105 (31, 25, 25, 23, 1)	5/104 (30, 25, 25, 23, 1)
	200	5/189 (18, 40, 36, 48, 47)	5/190 (18, 47, 36, 41, 28)
	300	6/329 (29, 64, 56, 71, 75, 34)	6/329 (24, 63, 57, 81, 81, 26)
	400	6/511 (22, 106, 65, 103, 121, 94)	6/560 (27, 100, 89, 105, 142, 97)
	500	7/* (28, 140, 127, >1000, 167, 198, 65)	7/893 (27, 142, 124, 137, 213, 203, 47)
15×15	100	5/195 (54, 46, 40, 25, 30)	5/216 (56, 41, 39, 72, 8)
	200	5/336 /(38, 87, 65, 67, 74, 5)	5/336 (38, 82, 66, 70, 75, 5)
	300	6/527 (44, 107, 89, 115, 103, 69)	6/522 (39, 108, 91, 113, 104, 67)
	400	7/811 (45, 147, 113, 172, 158, 160, 16)	7/842 (45, 150, 114, 173, 157, 168, 35)
	500	8/1340 (52, 252, 173, 207, 256, 194, 201, 5)	8/1262 (54, 244, 174, 205, 238, 135, 201, 11)

formulation and the CGSTAB iterative solver. This is not surprising, considering Carey's preconditioning matrix.

In Figure 5 three-dimensional cavity flow is shown for Reynolds number 400 and an $8 \times 8 \times 8$ grid. These results indicate that the preconditioner and CGSTAB algorithm also work in three dimensions. In Table VII the number of Newton iterations and CGSTAB iterations at each Newton step are shown for the three-dimensional Navier–Stokes equations. In order to achieve convergence with CGSTAB for high Reynolds numbers, the grid had to be correspondingly refined. To obtain convergence for Reynolds number 400, the grid has to be at least $8 \times 8 \times 8$ and to obtain convergence for Reynolds number 500, the grid has to be at least $10 \times 10 \times 10$. When the

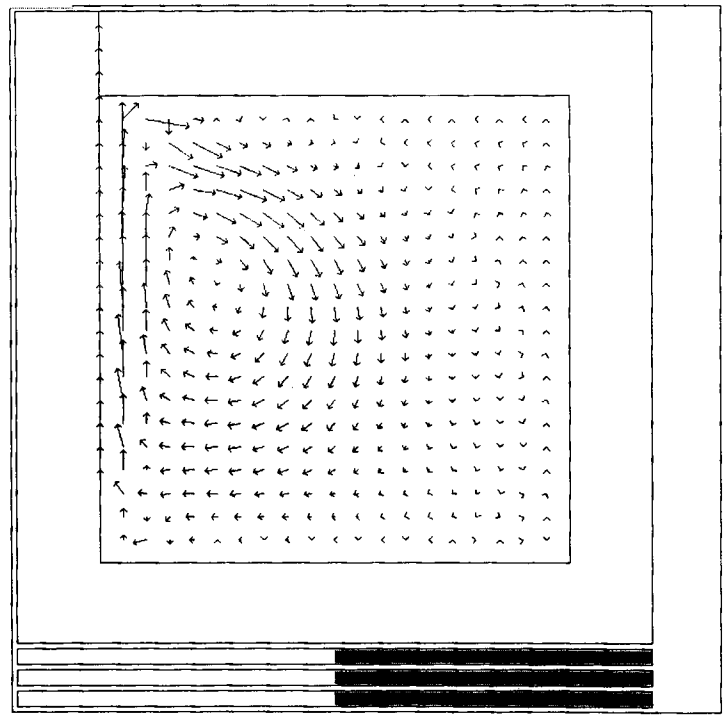


Figure 4. The solution of the Navier–Stokes problem for driven cavity flow in terms of velocity vectors for a 10×10 grid and Reynolds number 400

Table VI. The number of CGSTAB iterations with ILU^a preconditioning with varying maximum number of CGSTAB iterations for each Newton step. The Reynolds number is 400

Grid	Itmax	CGSTAB iterations
10×10	100	6/515 (26, 100, 89, 100, 100, 100)
	50	8/376 (26, 50, 50, 50, 50, 50, 50, 50)
	25	15/375 (25, . . .)
	10	17/170 (10, . . .)
15×15	100	9/836 (45, 100, 100, 100, 100, 100, 100, 100, 91)
	50	11/545 (45, 50, 50, 50, 50, 50, 50, 50, 50, 50, 50)
	25	>40/(25, . . .)
	10	>40/(10, . . .)

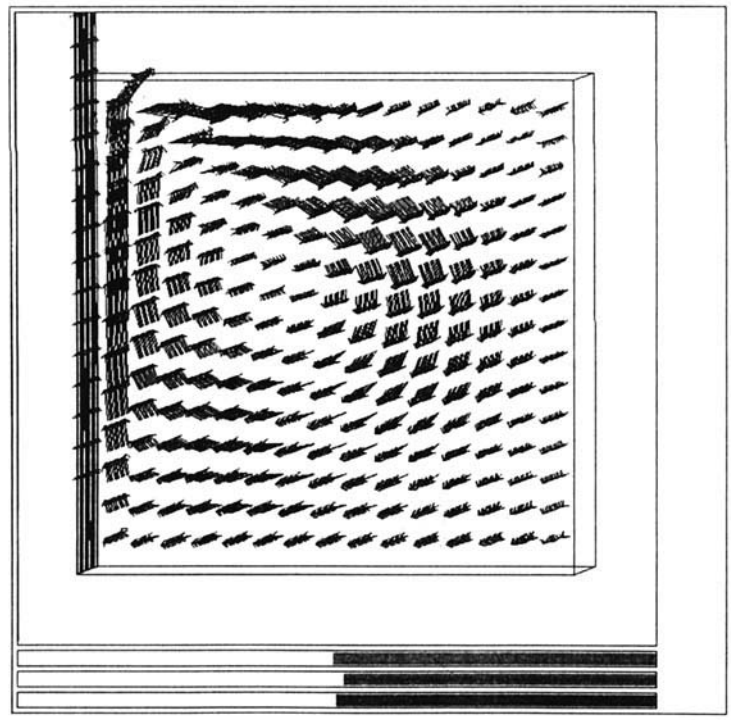


Figure 5. The solution of the Navier–Stokes equations for three-dimensional driven cavity flow in terms of velocity vectors for $8 \times 8 \times 8$ grid and Reynolds number 400

Table VII. The number of iterations for CGSTAB with ILU^s preconditioning for Navier–Stokes equations in three-dimensional cavity flow. No inner iterations are used. In the last column, the first number in the table is the number of Newton iterations, the second is the total number of CGSTAB iterations. The numbers in brackets are the number of CGSTAB iterations at each Newton step. The grid with resolution $10 \times 10 \times 10$ has 29 114 unknowns

Grid	*Re*	ILU^s
$5 \times 5 \times 5$	100	4/55 (11, 15, 15, 14)
	200	5/127 (13, 29, 29, 34, 22)
	300	6/339 (17, 63, 77, 75, 71, 36)
$8 \times 8 \times 8$	100	5/96 (19, 25, 25, 26, 1)
	200	5/205 (18, 59, 46, 46, 36)
	300	6/485 (23, 108, 91, 122, 119, 22)
	400	6/961 (31, 181, 182, 219, 199, 149)
$10 \times 10 \times 10$	100	5/135 (30, 37, 31, 33, 4)
	200	5/273 (27, 66, 60, 70, 50)
	300	6/616 (30, 150, 118, 145, 141, 32)
	400	7/1171 (29, 273, 187, 216, 232, 188, 46)
	500	7/1910 (40, 429, 260, 281, 319, 488, 93)

grid is refined, the element size is decreased with a reduction of the degree of non-symmetry in the equations. Since an 'upwind' finite element method has not been used, the equation matrices are expected to have complex eigenvalues with a large imaginary part, when the spatial grid is too coarse. The convergence properties of CGSTAB may be sensitive to such eigenvalues and thereby result in a slower convergence rate or no convergence at all.

DISCUSSION

In the present work, several preconditioners and two linear equation solvers have been investigated for solving the Stokes and Navier–Stokes equations with mixed finite element formulation. Concerning the linear equation solvers, the CGSTAB algorithm is clearly superior to the Orthomin algorithm. CGSTAB converged properly in several cases where Orthomin failed to converge or became stagnant. In some stagnant cases a restart of the Orthomin algorithm helped.

Among the preconditioners investigated, the most efficient one was incomplete factorization of the symmetric part of the Stokes equations with the allowance of slight fill-in. The numerical experiments show that both the incomplete factorization with slight fill-in and the CGSTAB iterative solver also work well in three-dimensional problems.

The results of the present experiments are compared to those obtained by Carey *et al.*[15] Carey *et al.* used a penalty formulation with full factorization of their Stokes equations as preconditioner and Orthomin (5) as linear equation solver. The equation matrix of the penalty formulation is very ill-conditioned, which probably leads to a slower convergence rate. On the other hand, the equation system for the same problem with mixed formulation requires more storage space as the pressure degrees of freedom are not eliminated from the equation system.

Comparison with Carey's results shows that the present method of solving the Navier–Stokes equations might be more efficient. Much less work in each linear iteration is needed with the present method since only incomplete factorization of the Stokes equations is used for preconditioning. In most cases both the number of Newton iterations as well as linear iterations are comparable.

For the non-linear Navier–Stokes equations different preconditioning matrices have been investigated. The best approach seems to use the symmetric part of the linear Stokes equation matrix as preconditioning matrix for all Newton steps.

Among the preconditioners investigated, incomplete LU factorization, ILU^s, of the symmetric part of the Stokes matrix with fill-in for coupled nodes in the pressure matrix is clearly the most efficient. This preconditioner permits natural fill-in where the original local element matrices contained a zero. The fill-in algorithm allows fill-in at predefined places in the equation matrix and needs no additional administration.

At present, the methods described in this paper seem efficient for computation of three-dimensional flow. This algorithm will be used together with the tri-tree unstructured grid generation algorithm[9] for investigation of transport of pollutants at sea-fjord interfaces in the near future.

ACKNOWLEDGEMENTS

The authors are grateful to Per Grøttum and Knut Liestøl for their comments on the manuscript. Otto Milvang and Jens Thomassen have been very helpful concerning the computer facilities. The authors are grateful to Harald Svendsen for valuable discussions. The authors are indebted to Lars Walløe for raising funds to make the project possible. The project has been supported by the Norwegian Research Council for Fisheries.

APPENDIX

The two Velocity Coupling and Pressure Coupling routines decide if two arbitrary nodes, and an arbitrary node and a corner node, respectively, are coupled. They return an index to where the second node (jj) is found in the array which contains the nodes in each row. If the node is not found, the return value is zero.

```
Velocity Coupling (ii, jj)
  if (PAC_ii = = PAC_jj) return(PAC_ii);
  if (ii gt jj)
  {
    for (k = PAC_jj; k < PAC_jj+1; k++)
    if (PAR_k = = ii) return(k);
  }
  if ( jj gt ii)
  {
    for (k = PAC_ii; k < PAC_ii+1; k++)
    if (PAR_k = = ii) return(k);
  }
  return(0);
  }
Pressure Coupling(ii, jj)
{
  for (k = PBC_ii; k < PBC_ii+1; k++)
  if(PBR_k = = jj) return(k);
  return(0);
}
```

The forward elimination of the matrix **A**

```
for (k = 1; k ≤ no. of nodes; k++)
{
  for (ii = PAC_k + 1; ii < PAC_k+1; ii++)
  for (jj = PAC_k + 1; jj < PAC_k+1; jj++)
  {i = PAR_ii; j = PAR_jj;
    if Velocity Coupling(i, j)
    {       if (i ≤ j)
      U_ij = U_ij − L_jk U_kk^-1 U_ki
      if (i > j)
      L_ij = L_ij − L_jk U_kk^-1 U_ki
      }
  }
}
```

The forward elimination of the matrix **B**

```
for (k = 1; k ≤ no. of nodes; k++)
{
  for (ii = PAC_k + 1; ii < PAC_k+1; ii++)
```

```
    for (jj = PBC_k + 1; jj < PBC_{k+1}; jj++)
    {i = PAR_ii; j = PBR_jj;
      if Pressure Coupling(i, j)
      B_ij = B_ij − L_ik U_kk^{-1} B_kj
    }
  }
```

The forward elimination of the matrix **C**

```
  for (k = 1; k ≤ no. of nodes; k++)
    for (ii = PBC_k + 1; ii < PBC_{k+1}; ii++)
    for (jj = PAC_k + 1; jj < PAR_{k+1}; jj++)
    {i = PBR_ii; j = PAR_jj;
              if Pressure Coupling(i, j)
              C_ij = C_ij − C_ik U_kk^{-1} U_kj
        }
      }
```

The fill in introduced to the matrix **P**

```
  for (k = 1; k ≤ no. of nodes; k++)
  {
    for (ii = PBC_k + 1; ii < PBC_{k+1}; ii++)
    for (jj = PBC_k + 1; jj < PBC_{k+1}; jj++)
    {i = PBR_ii; j = PBR_jj;
      if Pressure Coupling(i, j)
      P_ij = P_ij − C_ik U_kk^{-1} B_kj
    }
  }
```

The forward elimination of the matrix **P**

```
  for (k = 1; k ≤ no. of corner nodes; k++)
  {p = PBC_k; while (p! = PBR_k)p++;
    for (ii = p + 1; ii < PBC_{k+1}; ii++)
    for (jj = p + 1; jj < PBR_{k+1}; jj++)
    {i = PBR_ii; j = PBR_jj;
      if Pressure Coupling(i, j)
      P_ij = P_ij − P_ik P_kk^{-1} P_kj
    }
  }
```

The forward elimination algorithm of the right-hand side with the matrices **A** and **C**

```
  for (k = 1; k ≤ no. of nodes; k++)
  {
    for (jj = PAC_k + 1; jj < PAC_{k+1}; jj++)
    {j = PAR_jj
      b_vj = b_vj − L_jk U_kk^{-1} b_vk
    }
```

```
    for (jj = PBC_k + 1; jj < PBC_{k+1}; jj++)
    {j = PBR_jj
      b_{p_j} = b_{p_j} - C_jk U_kk^{-1} b_{v_k}
    }
  }
```

The forward elimination algorithm of the right-hand side with the matrix **P**

```
  for (k = 1; k ≤ no. of corner nodes; k++)
  {p = PBC_k; while (k! = PBR_p) p++;
    for (jj = p + 1; jj < PBC_{k+1}; jj++)
    {j = PAR_jj;
      b_{p_j} = b_{p_j} - P_jk U_kk^{-1} b_{p_k}
    }
  }
```

The backward substitution algorithm with the matrix **P**

```
  for (k = no. of corner nodes; k ≥ 1; k--)
  {p = PBC_k; while (k! = PBR_p) p++;
    for (jj = p + 1; jj < PBC_{k+1}; jj++)
    {j = PAR_jj;
      b_{p_k} = b_k - P_kj b_{p_j}
    }
    b_{p_k} = P_kk^{-1} b_k
  }
```

The subtraction of $\mathbf{B}\mathbf{b}_p$ from the right-hand side

```
  for (k = 1; k ≤ no. of nodes; k++)
  {
    for (jj = PBC_k + 1; jj < PBC_{k+1}; jj++)
    {j = PBR_jj;
      b_{v_k} = b_{v_k} - B_kj b_{p_j}
    }
  }
```

The backward substitution algorithm with the matrix **A**

```
  for (k = no. of nodes; k ≥ 1; k--)
  {
    for (jj = PAC_k + 1; jj < PAC_{k+1}; jj++)
    (j = PAR_jj;
      b_{v_k} = b_{v_k} - U_kj b_{v_j}
    }
    b_{v_k} = U_kk^{-1} b_{v_k}
  }
```

The approximate solution of $\mathbf{Q}\mathbf{x} = \mathbf{b}$ is contained in **b**.

REFERENCES

1. T. Utnes, 'Finite element modeling of quasi-three dimensional nearly horizontal flow', *Int. j. numer. methods fluids*, **12**, 559–576 (1991).
2. G. Furnes, 'A three-dimensional numerical sea model with eddy viscosity varying piecewise linearly in the vertical', *Continental Shelf Res.*, **2**, 231–241 (1983).
3. R. R. Tharera, J. R. Stewart, O. Hassan, K. Morgan and J. Peraire, 'A point implicit unstructured grid solver for the Euler and Navier–Stokes equations', *AIAA 26th Aerospace Sciences Meeting*, Reno, Nevada, 1988.
4. M. Jayaram and A. Jameson, 'An efficient vector and parallel multigrid method for viscous transonic flow', *Proc. Int. Symp. on High Performance Computing*. Montpellier, France, 1989, pp. 225–237.
5. T. J. R. Hughes, L. P. Franca and G. M. Hulbert, 'A new finite element formulation for computational fluid dynamics: VIII. The Galerkin/least-squares method for advective–diffusive equations', *Comput. Meths. Appl. Mech. Eng.*, **73**, 173–189 (1989).
6. S. Ø. Wille, 'Numerical simulations of steady flow inside a three dimensional aortic bifurcation model', *J. Biomed. Eng.*, **6**, 49–55 (1984).
7. S. Ø. Wille and L. Walløe, 'Pulsatile pressure and flow in arterial stenosis simulated in a mathematical model', *J. Biomed. Eng.*, **3**, 17–24 (1981).
8. S. Ø. Wille, 'Pulsatile pressure and flow in arterial aneurysm simulated in a mathematical model', *J. Biomed. Eng.*, **3**, 153–158 (1981).
9. S. Ø. Wille, 'A structured tri-tree search method for generation of optimal unstructured finite element grids in two and three dimensions', *Int. j. numer. methods fluids*, **14**, 861–881 (1992).
10. I. Gustafsson, 'A class of first order factorization methods', *BIT*, **8**, 142–156 (1978).
11. E. Barragy and G. F. Carey, 'A partitioning scheme and iterative solution for sparse bordered systems', *Comput. Meths. Appl. Mech. Eng.*, **70**, 321–327 (1988).
12. H. P. Langtangen, T. Rusten, A. Tveito and S. Ø. Wille, 'An element by element preconditioner for iterative equation solvers', *Proc. 6th Int. Conf. on Finite Elements in Water Resources*. Lisbon, Portugal, 1986.
13. S. Ø. Wille, 'An element by element preconditioner for refined finite element grids', *Proc. 1st Int. Conf. on Numerical Grid Generation in Computational Fluid Dynamics*, Landshut, Germany, 1986.
14. S. Ø. Wille, 'Local gaussian preconditioning of symmetric and non symmetric finite element equation solvers', *Proc. 1st Int. Conf. Industrial and Applied Mathematics, ICIAM 87*. Paris, France, 1987.
15. G. F. Carey, K. C. Wang and W. D. Joubert, 'Performance of iterative methods for Newtonian and generalized Newtonian flows', *Int. j. numer. methods fluids*, **9**, 127–150 (1989).
16. J. A. Meijerink and H. A. van der Vorst, 'An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix', *Math. Comp.*, **31**, 148–162 (1977).
17. D. M. Young and K. C. Yea, 'Generalized conjugate-gradient acceleration of non-symmetrizable iterative methods', *Lin. Alg. and its Applications*, **34**, 159–194 (1980).
18. O. G. Johnson, C. A. Michelli and G. Paul, 'Polynomial preconditioners for conjugate gradient calculations', *SIAM J. Numer. Anal.*, **20**, 362–376 (1983).
19. P. K. W. Vinsome, 'Orthomin, an iterative method for solving sparse sets of simultaneous linear equations', *Proc. 4th Symp. Reservoir Simulation*, Society of Petroleum Engineers of AIME, 1976, pp. 147–159.
20. P. Sonneveld, 'CGS, a fast Lanczos-type solver for nonsymmetric linear systems', *SIAM J. Stat. Comput.*, **10**, 36–52 (1987).
21. H. A. van der Vorst and P. Sonneveld, 'A more smoothly converging variant of CG-S'. *Report 90-50*. Faculty of Technical Mathematics and Informatics, Delft University of Technology, The Netherlands. ISSN 0922-5641 (1990).
22. H. A. van der Vorst, 'Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems', *SIAM J. Sci. Stat. Comp.* (to appear).
23. D. Howard, W. M. Connolley and J. S. Rollett, 'Unsymmetric conjugate gradient methods and sparse direct methods in finite element flow simulation', *Int. j. numer. methods fluids*, **10**, 925–945 (1990).
24. Taylor C. and P. Hood, 'A numerical solution of the Navier–Stokes equations using the finite element technique', *Computers and Fluids*, **1**, 73–100 (1973).
25. A. George and J. W. Liu, *Computer Solutions of Large Sparse Positive Definite Systems*, Prentice-Hall, Englewood Cliffs, NJ, 1981.