INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2008; **57**:1731–1751
Published online 17 December 2007 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1684

A comparison of preconditioners for incompressible Navier–Stokes solvers

M. ur Rehman, C. Vuik*,† and G. Segal

Faculty EEMCS, Delft Institute of Applied Mathematics, Delft University of Technology, 07.060, Mekelweg 4, 2628 CD, Delft, The Netherlands

SUMMARY

We consider solution methods for large systems of linear equations that arise from the finite element discretization of the incompressible Navier–Stokes equations. These systems are of the so-called saddle point type, which means that there is a large block of zeros on the main diagonal. To solve these types of systems efficiently, several block preconditioners have been published. These types of preconditioners require adaptation of standard finite element packages. The alternative is to apply a standard ILU preconditioner in combination with a suitable renumbering of unknowns. We introduce a reordering technique for the degrees of freedom that makes the application of ILU relatively fast. We compare the performance of this technique with some block preconditioners. The performance appears to depend on grid size, Reynolds number and quality of the mesh. For medium-sized problems, which are of practical interest, we show that the reordering technique is competitive with the block preconditioners. Its simple implementation makes it worthwhile to implement it in the standard finite element method software. Copyright © 2007 John Wiley & Sons, Ltd.

Received 14 December 2006; Revised 10 October 2007; Accepted 14 October 2007

KEY WORDS: preconditioners; incompressible Navier–Stokes; saddle point; Krylov method; block preconditioner; ILU preconditioner

1. INTRODUCTION

In the past few decades, computational methods for solving the Navier–Stokes equations have proven to be very useful for obtaining better understanding of the physics of flows. The increase in computing power has driven the advances in algorithm development. Nowadays, it is possible

*Correspondence to: C. Vuik, Faculty EEMCS, Delft Institute of Applied Mathematics, Delft University of Technology, 07.060, Mekelweg 4, 2628 CD, Delft, The Netherlands.
†E-mail: c.vuik@tudelft.nl

Contract/grant sponsor: Higher Education Commission of Pakistan and its Dutch handling agent Nuffic; contract/grant number: 2-17/PM-OS/Neth/2005

to apply the finite element method (FEM) to large 3D problems. Most of the researches these days are focused on the efficient solution of these equations on complex domains. Discretization of the Navier–Stokes equations leads to a large system of non-linear equations. Linearization methods reduce this to a series of large systems of linear equations. These systems can be solved by direct methods; however, doing that requires extensive resources in terms of computing time and memory. For 3D and large 2D problems, iterative methods, in combination with a suitable preconditioner, are the method of choice.

Most of the recent papers on the iterative solution of the discretized Navier–Stokes equations are devoted to *block preconditioners* [1–5]. Classical ILU preconditioners in combination with a suitable node-renumbering technique [6–9] have been applied with success.

ILU preconditioners require less adaptation of standard finite element packages than block preconditioners and are therefore easier to implement. In this paper we shall derive an *a priori* ordering of unknowns that, in combination with a node-renumbering technique, improves the convergence behavior of ILU preconditioners. It is shown that this combination is an interesting alternative to block preconditioners.

In Section 2 the discretization of the incompressible Navier–Stokes by an FEM is considered. The resulting equations are linearized by the Picard or the Newton method. This leads to a system of linear equations with a large number of zeros on the main diagonal due to the absence of the pressure term in the continuity equation. Systems of equations of this form are singular if the discretization of the continuity equation leads to an underdetermined system. To avoid this problem, the finite elements must satisfy the so-called Brezzi–Babuška (BB) condition. An alternative is to apply a stabilization technique [10–12].

The iterative solution of these systems of linear equations requires suitable preconditioners. An overview of recent block preconditioners is given in Section 3.

Direct solution of the coupled discrete linearized Navier–Stokes equations may fail due to zeros on the main diagonal unless partial pivoting is applied. The same may happen in the case of ILU factorization. Wille and coworkers [13–15] use a node-renumbering technique in combination with a reordering of unknowns that is common for block preconditioners. This method can be applied to both direct and ILU-preconditioned iterative methods. In Section 4 we present a modification of this technique. We define a reordering of unknowns that leads to an almost optimal profile or bandwidth for a direct solver. Applied to an ILU preconditioner, this reordering usually improves the convergence behavior of Krylov subspace methods. The method can be applied in combination with any node renumbering strategy. Numerical experiments are presented in Section 5. Results for the benchmark problem of the backward facing step, both in 2D and in 3D, show that, despite being relatively simple, our method performs remarkably well. Finally, in Section 6 we present our conclusions.

2. INCOMPRESSIBLE NAVIER–STOKES EQUATIONS

We consider the steady-state Navier–Stokes equations for the flow of a Newtonian incompressible viscous fluid with constant viscosity:

$$-\nu\nabla^2\mathbf{u}+\mathbf{u}\cdot\nabla\mathbf{u}+\nabla p=\mathbf{f} \quad \text{in } \Omega \tag{1}$$

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

where Ω is a 2D or 3D domain with a piecewise smooth boundary $\partial\Omega$, $\mathbf{u}$ is the fluid velocity, p is the pressure field, $\nu>0$ is the kinematic viscosity coefficient (inversely proportional to the Reynolds number Re), and ∇ is the gradient and $\nabla\cdot$ is the divergence operator.

Equation (1) represents conservation of momentum, whereas Equation (2) represents mass conservation (incompressibility condition). The boundary value problem that we consider is systems (1) and (2) posed on Ω together with boundary conditions on $\partial\Omega=\partial\Omega_E\cup\partial\Omega_N$ and is given by

$$\mathbf{u}=\mathbf{w} \quad \text{on } \partial\Omega_E, \quad \nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}}-\mathbf{n}p=0 \quad \text{on } \partial\Omega_N$$

The discretization of the Navier–Stokes equations is done through the FEM. The weak formulation of the Navier–Stokes equations is given by finding $\mathbf{u}\in\mathbf{H}^1_E$ and $p\in L_2(\Omega)$ such that

$$\nu\int_\Omega \nabla\mathbf{u}:\nabla\mathbf{v}\,\mathrm{d}\Omega+\int_\Omega(\mathbf{u}\nabla\cdot\mathbf{u})\cdot\mathbf{v}\,\mathrm{d}\Omega+\int_\Omega p\nabla\cdot\mathbf{v}\,\mathrm{d}\Omega=\int_\Omega\mathbf{f}\cdot\mathbf{v}\,\mathrm{d}\Omega \quad \forall\mathbf{v}\in\mathbf{H}^1_{E_0} \tag{3}$$

$$\int_\Omega(\nabla\cdot\mathbf{u})q\,\mathrm{d}\Omega=0 \quad \forall q\in L_2(\Omega) \tag{4}$$

where $\mathbf{H}^1_E$ is the Sobolev space of functions satisfying the essential boundary conditions, $\mathbf{H}^1_{E_0}$ is the Sobolev space satisfying the homogeneous essential boundary conditions, and : denotes the dyadic product.

For a discrete weak formulation, we define finite dimensional subspaces $\mathbf{X}^h_0\subset\mathbf{H}^1_{E_0}$, $M^h\subset L_2(\Omega)$, and $\mathbf{X}^h_E\subset\mathbf{H}^1_E$. The discrete version of (3) and (4) is finding $u_h\in\mathbf{X}^h_E$ and $p_h\in M^h$ such that

$$\nu\int_\Omega\nabla\mathbf{u_h}:\nabla\mathbf{v_h}\,\mathrm{d}\Omega+\int_\Omega(\mathbf{u_h}\cdot\nabla\mathbf{u_h})\cdot\mathbf{v_h}\,\mathrm{d}\Omega-\int_\Omega p_h(\nabla\cdot\mathbf{v_h})\,\mathrm{d}\Omega=\int_\Omega\mathbf{f}\cdot\mathbf{v_h}\,\mathrm{d}\Omega \quad \forall\mathbf{v_h}\in\mathbf{X}^h_0 \tag{5}$$

$$\int_\Omega(\nabla\cdot\mathbf{u_h})q_h\,\mathrm{d}\Omega=0 \quad \forall q_h\in M^h \tag{6}$$

Formally, the system of non-linear equations can be expressed as

$$A\mathbf{u}+N(\mathbf{u})+B^{\mathrm{T}}p=\mathbf{f} \tag{7}$$

$$B\mathbf{u}=0 \tag{8}$$

where $A\mathbf{u}$ is the discretization of the viscous term, $N(\mathbf{u})$ is the discretization of the non-linear convective term, $B\mathbf{u}$ denotes the discretization of the negative divergence of $\mathbf{u}$, and $B^{\mathrm{T}}p$ is the discretization of the gradient of p. The right-hand side vector, $\mathbf{f}$, contains all the contributions of the source term, the boundary integral, as well as the contribution of the prescribed boundary conditions.

Systems of the form (7) and (8) are called saddle point problems. Owing to the absence of the pressure term in Equation (8), the system of equations may be underdetermined for an arbitrary combination of pressure and velocity unknowns. In order to guarantee a unique solution of Equations (7) and (8), the finite element discretization should satisfy the BB condition given as

$$\inf_{q_h\in M^h}\sup_{v_h\in X^h_0}\frac{(\nabla\cdot v_h,q_h)}{\|v_h\|_{X^h_0}\|q_h\|_{M^h}}\geqslant\gamma>0 \tag{9}$$

If a finite element does not satisfy the BB condition, solution of (7) and (8) is possible only by applying stabilization techniques [5]. In our experiments we have limited ourselves to finite elements that satisfy the BB condition.

In 2D we used the $Q2$–$Q1$, Taylor–Hood [16], element (continuous pressure) and the $Q2$–$P1$, Crouzeix–Raviart [17], element with discontinuous pressure. For details see [18]. In 3D we used a triquadratic velocity field with trilinear pressure Taylor–Hood element, denoted as $Q2$–$Q1$. Also a triquadratic velocity with discontinuous linear pressure Crouzeix–Raviart element is applied (denoted as $Q2$–$P1$).

In order to solve the non-linear equations (7) and (8), they are linearized and combined with some iteration process. Picard and Newton methods are coupled with exact or inexact linear solvers. After linearization, the system can be expressed as

$$\begin{bmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{10}$$

where F contains the contributions from the linearized convection diffusion term. The system matrix of (10) is indefinite and non-symmetric.

For small-sized problems, direct solvers are very efficient to solve the system given in (10), but they are not feasible for large problems because they require large central processing unit (CPU) time and a huge memory. Preconditioned Krylov solvers are good alternatives for direct solvers in large problems. Preconditioned GMRES [19] and Bi-CGSTAB [20] are popular methods to solve the system. Some results are given with the GMRESR method [21], which shows better convergence, but can be more expensive than GMRES and Bi-CGSTAB.

3. BLOCK PRECONDITIONERS FOR THE NAVIER–STOKES EQUATIONS

Firstly, the objective is to design a preconditioner for the Navier–Stokes equations, which enhances the convergence of an iterative method independently of the Reynolds number and number of grid points. Secondly, the application of a preconditioner should be cheap. Block triangular preconditioners are a well-known class of preconditioners that exploit the block structure of the Navier–Stokes problem. The block triangular preconditioner is expressed as

$$P = \begin{bmatrix} F & B^{\mathrm{T}} \\ 0 & S \end{bmatrix} \tag{11}$$

where $S = -BF^{-1}B^{\mathrm{T}}$ is the Schur complement matrix. The steps involved in a block triangular preconditioner to solve the system $Pz = r$ are

$$\text{solve } z_2 \text{ from } Sz_2 = r_2 \text{ and } z_1 \text{ from } Fz_1 = r_1 - B^{\mathrm{T}} z_2 \tag{12}$$

The preconditioner involves the solution of two subproblems associated with the velocity and the pressure part. An eigenvalue analysis of the block triangular preconditioner in (11) suggests that GMRES converges in two iterations [22] if exact arithmetic is used. But in general, F^{-1} and S^{-1} are very expensive to compute and to store. Usually, F^{-1} is formally approximated by a matrix $\hat{F}^{-1}$. Actually, such an approximation consists of a small number of iterations with an iterative method. The approximation of the Schur complement is problem dependent. In general, the Schur complement matrix is not formed. Therefore, the approximate inverse, $\hat{S}^{-1}$, is replaced by a simple spectrally equivalent matrix. Various cheap approximations of S^{-1} have been published recently. For an overview of preconditioners, we refer to [1, 2, 5, 23]. Some of those that are used in combination with the block triangular preconditioner (11) are discussed below.

3.1. Pressure convection diffusion (PCD)

A popular approximation to the Schur complement, known as PCD, is given by Kay *et al.* [3] and Silvester *et al.* [24]. The approximation is based on a discrete convection diffusion operator. The operator on the velocity space is defined as

$$\mathscr{L} = -\nu\nabla^2 + \mathbf{w_h}\cdot\nabla \tag{13}$$

where $\mathbf{w_h}$ is the approximation to the discrete velocity computed in the most recent Picard iteration. Suppose that the commutator of the convection diffusion operator on the velocity space, multiplied by the gradient operator on the velocity space, and the gradient operator acting on the convection diffusion operator in the pressure space are small. Hence

$$\varepsilon = (-\nu\nabla^2 + \mathbf{w_h}\cdot\nabla)\nabla - \nabla(-\nu\nabla^2 + \mathbf{w_h}\cdot\nabla)_\mathrm{p} \tag{14}$$

is small. Then the discrete commutator in terms of finite element matrices given as

$$\varepsilon_h = (Q^{-1}F)(Q^{-1}B^\mathrm{T}) - (Q^{-1}B^\mathrm{T})(Q_\mathrm{p}^{-1}F_\mathrm{p}) \tag{15}$$

is also small. F_p is a discrete convection–diffusion operator on pressure space. Q denotes the velocity mass matrix and Q_p the pressure mass matrix. The multiplication by Q^{-1} and Q_p^{-1} transforms quantities from integrated values to nodal values. Pre-multiplication of (15) by $BF^{-1}Q$, post-multiplication by $F_\mathrm{p}^{-1}Q_\mathrm{p}$, and assuming that the commutator is small lead to the Schur approximation:

$$BF^{-1}B^\mathrm{T} \approx BQ^{-1}B^\mathrm{T}F_\mathrm{p}^{-1}Q_\mathrm{p} \tag{16}$$

The expensive part $BQ^{-1}B^\mathrm{T}$ in (16) is replaced by its spectrally equivalent matrix A_p known as the pressure Laplacian matrix; hence

$$S = -BF^{-1}B^\mathrm{T} \approx -A_\mathrm{p}F_\mathrm{p}^{-1}Q_\mathrm{p} \tag{17}$$

The preconditioner is known as the PCD preconditioner. This preconditioner is also effective for stabilized finite elements. The preconditioner has nice convergence properties especially for enclosed flows if the equations are linearized by the Picard method. The preconditioner requires the action of a Poisson solve, a mass matrix solve, and a matrix–vector product with F_p. Moreover, boundary conditions should also be taken into account while constructing A_p and F_p.

3.2. Least-squares commutator (LSC)

Another promising approach for the approximation of the Schur complement is described by Elman *et al.* [25] and is known as an LSC preconditioner. Instead of deriving the relation for the Schur complement, an approximation is made to the matrix operator, F_p, in (16), that makes the commutator (15) small. This can be achieved by solving a least-squares problem. For the jth column of matrix F_{p}, the least-squares problem is of the form:

$$\min \|[Q^{-1}FQ^{-1}B^{\mathrm{T}}]_j - Q^{-1}B^{\mathrm{T}}Q_{\mathrm{p}}^{-1}[F_{\mathrm{p}}]_j\|_Q \tag{18}$$

where $\|.\|_Q$ is the $\sqrt{\underline{x}^{\mathrm{T}}Q\underline{x}}$ norm. The normal equations associated with this problem are

$$Q_{\mathrm{p}}^{-1}BQ^{-1}B^{\mathrm{T}}Q_{\mathrm{p}}^{-1}[F_{\mathrm{p}}]_j = [Q_{\mathrm{p}}^{-1}BQ^{-1}FQ^{-1}B^{\mathrm{T}}]_j$$

which leads to the following definition of F_{p}:

$$F_{\mathrm{p}} = Q_{\mathrm{p}}(BQ^{-1}B^{\mathrm{T}})^{-1}(BQ^{-1}FQ^{-1}B^{\mathrm{T}})$$

Substituting this expression into (16) gives an approximation of the Schur complement matrix:

$$BF^{-1}B^{\mathrm{T}} \approx (BQ^{-1}B^{\mathrm{T}})(BQ^{-1}FQ^{-1}B^{\mathrm{T}})^{-1}(BQ^{-1}B^{\mathrm{T}}) \tag{19}$$

Usually, the inverse of the velocity mass matrix Q^{-1} is dense. In such a case, the velocity mass matrix is replaced by the diagonal matrix $\hat{Q}$ (the main diagonal of Q). The preconditioner can be used only for stable elements. It does not require any extra operator as (17) used in PCD. Its convergence is in some cases better than that of PCD. LSC is more expensive than PCD because it requires two Poisson solves. The convergence of LSC is independent of the mesh size and mildly dependent on the Reynolds number. For more details on PCD and LSC, refer to [5].

3.3. Augmented Lagrangian (AL) approach

An effective preconditioner based on the AL approach is recently published by Benzi and Olshanskii [4]. This technique suggests to approximate the inverse Schur complement by

$$\hat{S}^{-1} = -(\nu\hat{Q}_p^{-1} + \gamma W^{-1}) \tag{20}$$

where $\hat{Q}_{\mathrm{p}}$ denotes the approximate pressure mass matrix, ν is the viscosity, and $\gamma>0$ is a parameter. Usually, W is also replaced by $\hat{Q}_{\mathrm{p}}$. For constant pressure approximation, Q_{p} is a diagonal matrix. For a linear pressure approximation, Q_{p} is replaced by the spectrally equivalent diagonal matrix. For a diagonal matrix $\hat{Q}_{\mathrm{p}}$, the calculation of the inverse approximate Schur complement is very cheap. For this preconditioner, the original system given in (10) is replaced by

$$\begin{bmatrix} F+\gamma B^{\mathrm{T}}W^{-1}B & B^{\mathrm{T}} \\ B & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{21}$$

Since $Bu=0$ we can add the term $\gamma B^{\mathrm{T}}W^{-1}Bu$ to the first row in (10) without any change on the right-hand side. Results given in [4] reveal that the convergence of an iterative method with this preconditioner is independent of the mesh size and Reynolds number and seems to be more robust than the other two schemes discussed above. The method is efficient if it is used as a preconditioner in the solution of the Navier–Stokes problem, linearized by Picard's iteration for both constant and linear pressure approximations. But this preconditioner requires an efficient method to solve the (1, 1) block, since (in our problem) the number of non-zeros increases due to the introduction of $\gamma B^{\mathrm{T}}W^{-1}B$. This matrix introduces a coupling between components of the velocity vector. A good choice of γ is also important. Usually γ is taken to be one.

In all the preconditioners discussed above, the S^{-1} and F^{-1} block matrices form the most expensive part of the preconditioner. The subsystems associated with these block matrices have to be solved efficiently, preferably by an iterative solver, for example, by geometric multigrid (MG) or algebraic multigrid (AMG). In this paper, we used an MG or a direct solver.

4. A SADDLE POINT ILU-TYPE PRECONDITIONER

The block preconditioners from the prior sections have the disadvantage that straightforward application of standard finite element codes is not possible. Adaptation of the matrix builder and solver is necessary since splitting of velocity and pressure unknowns is required. From a practical point of view, it would be attractive if standard classical iterative solution schemes, such as preconditioned Krylov solvers, could be applied without any changes. However, in the case of non-stabilized elements, the zero pressure block in the continuity equation prevents straightforward application of LU and ILU factorization. If the common ordering of unknowns is used, i.e. placing first all unknowns of node 1, then those of node 2 and so on, one might obtain a zero pivot, especially if velocities at some boundaries are prescribed and therefore both factorizations may fail. Pivoting [26], on the other hand, will result in a large increase in memory usage and as a consequence computation time. Besides that, it is hard to predict *a priori* the amount of memory required, which from an implementation point of view is not very practical. To avoid this problem, it is better to use a suitable *a priori* reordering of unknowns. As pointed out by Wille and others [13–15], pivoting is not necessary when the unknowns are ordered in the sequence, so that all velocity unknowns come first and then all the pressure unknowns like in the block preconditioners. The reason being that during (incomplete) factorization the zeros at the main diagonal will vanish, provided fill-in is allowed based on the connectivity of nodal points rather than actual zeros in the matrix. Renumbering of nodal points as suggested by Wille and also classical renumbering techniques as suggested by Cuthill McKee [27] and Sloan [28] may decrease the memory and computation time for direct solvers. An optimal numbering of unknowns, for a direct solver, usually improves the convergence of ILU preconditioned Krylov solvers [8, 9], but exceptions to this rule exist [1].

We propose a new *a priori* ordering of unknowns, which in combination with a suitable renumbering of nodes results in an optimal bandwidth or profile (envelope) of the coefficient matrix in the case of a direct method. We will demonstrate that the combination of this reordering technique with ILU preconditioned Krylov solvers results in better solver performance. Experiments in Section 5 show that, for practical 2D and 3D meshes, this method behaves very well compared with the block solvers of the previous section. The extra advantage is of course the simplicity of the implementation.

4.1. Ordering scheme for direct solvers

If, for a direct solver, we use the same ordering as in the case of the block preconditioners, i.e. placing first all velocity unknowns and then all pressure unknowns, we end up with a very large profile of the matrix. This is true even if we use an optimal node renumbering. The main advantage of this ordering is that no pivoting is necessary since during factorization the zeros on the main diagonal in the zero pressure block disappear, see, for example, [14]. In the remaining part of this paper, we shall refer to this reordering as *p-last*. Figure 2 shows an example of the

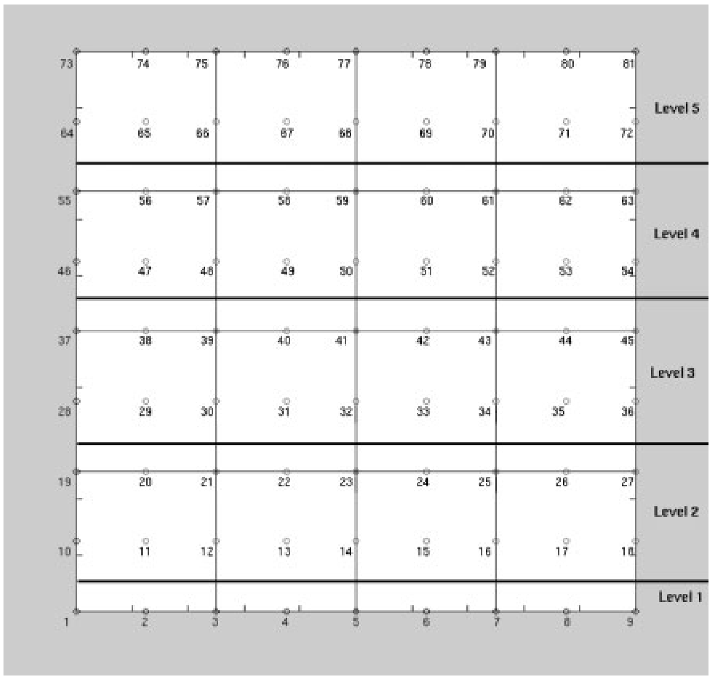

Figure 1. Levels defined for 4×4 $Q2$–$Q1$ grid.

non-zero structure of the matrix for the p-last ordering, applied to a 4×4 rectangular grid of $Q2$–$Q1$ elements, where a lexicographic numbering of nodes is used.

A much smaller profile will be achieved, if we order the unknowns in the sequence of the nodal points. However, especially in the case of Dirichlet boundary conditions for the velocity, this may lead to zero pivots. Hence, if we try to avoid pivoting during elimination, this ordering cannot be applied. In order to obtain a reordering that has almost the same favorable profile as the node-wise ordering but does not lead to zero pivots, we need to define the concept of levels, which originates from the classical Cuthill McKee renumbering scheme.

Let us first define the notion of levels for Cuthill McKee. Suppose that we have created levels 1 to $i-1$. Then level i is defined as the set of nodes that are connected directly to level $i-1$ and are not in one of the prior levels. Nodes are connected if they belong to the same element.

The first level may be defined as a point or even a line in R^2 or a surface in R^3. This definition also applies in the case of structured grids. For example, in the 4×4 structured grid of Figure 1 with $Q2$–$Q1$ elements, the first level might consist of the nodes 1–9. Level 2 consists of the nodes 10–29 and so on. It is clear that nodes in level i are connected only to nodes in levels $i-1$ and $i+1$.

In case of a different renumbering scheme like Sloan [28] or the one proposed by Wille *et al.* [14], we define levels in the following way.

Suppose that levels 1 to $i-1$ have been constructed. Let node k be the node with highest node number that is directly connected to nodes in level $i-1$. Then level i consists of node k and all nodes with node number less than k but not belonging to one of the levels 1 to $i-1$.

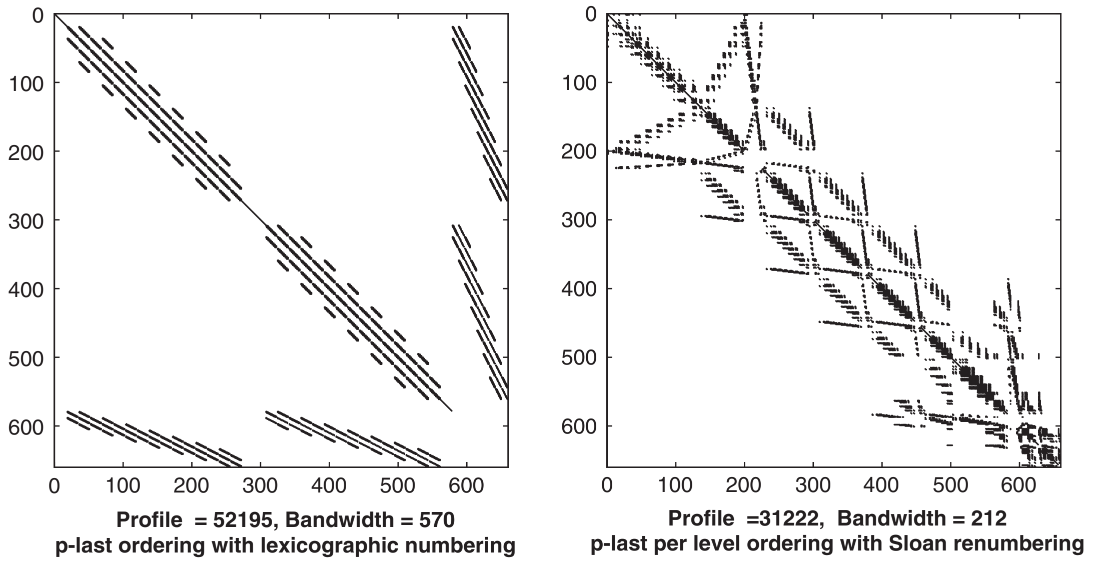

Figure 2. Effect of Sloan renumbering of grid points and p-last per level reordering of unknowns on the profile and bandwidth of the matrix.

The first level is defined as node 1.

Once the levels have been defined, we reorder the unknowns in the following way. First, we take all the velocities of level 1, then all pressures of level 1. Next, we do the same for level 2 and repeat this process for all levels. Hence, instead of a global block ordering, we apply a block ordering per level. Such an approach has two advantages.

First, the profile is hardly enlarged compared with the optimal ordering, since the local bandwidth is defined by the largest distance in node numbers.

Second, due to the local block reordering, zero pivots become non-zero during factorization, and no *a posteriori* pivoting is required. In the remainder we shall refer to this ordering technique as *p-last per level*.

One has to be careful at the start of this process. If, for example, the velocities in node 1 are prescribed, we start with a pressure unknown that gives rise to a zero pivot. Therefore, we always combine levels 1 and 2 into a new level. If the number of free velocity unknowns in this new level is less than the number of pressure unknowns, we also add the next level to level 1 and if necessary this process is repeated. In practice, a combination of 2 or 3 levels is sufficient. Note that the starting level has always a small contribution to the global profile. Figures 2 and 3 show the effect of the p-last per level renumbering, combined with Sloan and Cuthill McKee renumbering, respectively. The grid used consists of 8×8 $Q2$–$Q1$ elements. The gain in memory is clear even for this small example.

Hence, our reordering technique p-last per level in combination with a suitable node renumbering strategy produces a nearly optimal profile and avoids the need for a pivoting in the case of direct solvers. It has been applied to many practical problems without ever producing small pivots. Since optimal reordering of unknowns for direct methods is usually also suitable for ILU preconditioners, we consider this method in the next subsection.

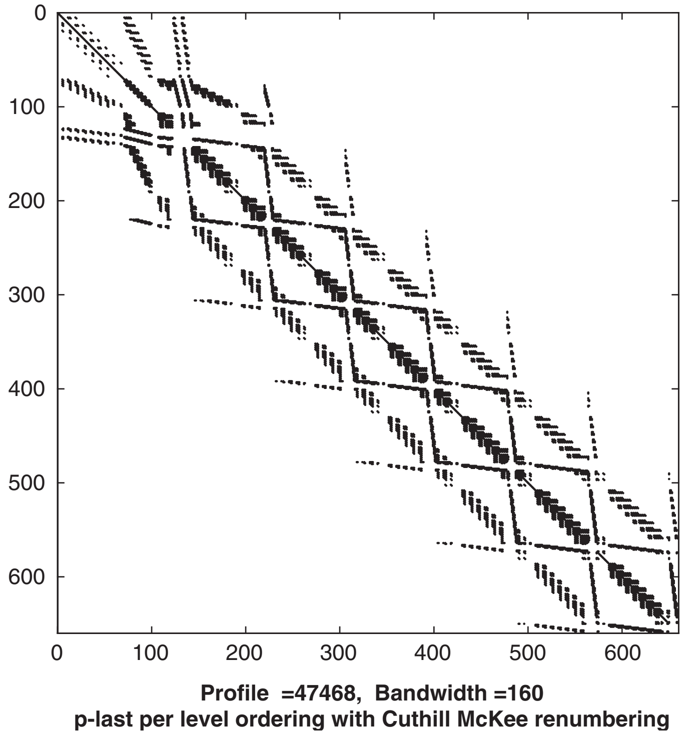

Figure 3. Effect of Cuthill McKee renumbering of grid points and p-last per level reordering of unknowns on the profile and bandwidth of the matrix.

4.2. Application to ILU preconditioning

The block preconditioners of Section 5 all require adaptation of standard finite element software packages. ILU-preconditioned Krylov subspace solvers, on the other hand, can be applied without any change at all. Since an optimal ordering of unknowns for a direct solver usually improves the behavior of an ILU preconditioner, we investigate p-last per level ordering as well as p-last ordering in combination with ILU.

We define the set S of fill-in positions as the set of unknowns that are directly connected. This implies that zeros in the pressure block may also be part of the set S, provided there is a connectivity with velocity unknowns. The ILU decomposition $A \approx LD^{-1}U$ is defined by the following rules:

$$\begin{aligned} l_{i,j} &= 0 \quad \text{for } (i,j) \notin S \\ u_{i,j} &= 0 \quad \text{for } (i,j) \notin S \\ (LD^{-1}U)_{i,j} &= a_{i,j} \quad \text{for } (i,j) \in S \end{aligned} \tag{22}$$

Experiments in Section 5 show that in a large number of practical cases this method performs very well and is competitive with the block preconditioners. However, in some cases the Krylov method converges slowly, or even diverges, for example, in the case of stretched grids using elements with a large aspect ratio. In that case we apply *extra fill-in* referred to as ILUF. Extra fill-in is defined by adding all neighbor points of the standard ILU node structure to the connectivity set, provided these nodes do not affect the envelope of the original matrix. In many cases extra fill-in solves the convergence problem, but at the cost of extra memory and computing time per iteration.

4.2.1. Lumping. Sometimes we apply lumping, that is, if the off-diagonal components of the matrix have the same sign as the diagonal of the matrix, these components are added to the diagonal of the matrix and made zero themselves. Of course, this is used to compute only the preconditioner. In our experience, the preconditioning matrix P_ℓ (lumped) improves convergence in some cases, but it should be used only if one does not achieve convergence with extra fill-in.

4.2.2. Perturbation of the continuity equation. In some cases, convergence can be improved by perturbing the continuity equation with a factor εp. The discretized form of the perturbed system of equations is given as

$$\begin{bmatrix} F & B^{\mathrm{T}} \\ B & \varepsilon Q_{\mathrm{p}} \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{23}$$

Hence, the incompressibility condition is violated by a small amount that hardly influences the solution. Note that this is almost the same as applying the penalty function method [18]. The only difference is that p is not eliminated from the second row in (23). Since, in general, it is hard to find a suitable value of ε, we have decided to use mostly $\varepsilon = 0$. In some cases, stretched grids and $Q2$–$P1$ discretization, we have perturbed the incompressibility constraints and found good convergence.

5. NUMERICAL EXPERIMENTS

In this section we present some numerical experiments. We start with an investigation of the properties of the saddle point ILU preconditioner. Thereafter, we compare this preconditioner with the block preconditioners as given in Section 3. The Stokes and the Navier–Stokes problems are solved in the following domains:

1. The Poisseuille channel flow in a square domain $(-1,1)^2$ with a parabolic inflow boundary condition and a natural outflow condition having the analytic solution: $u_x = 1 - y^2$, $u_y = 0$ and $p = 2\nu x$. In the case of Stokes flow $\nu = 1$.
2. The L-shaped domain $(-1, L) \times (-1, 1)$, known as the backward facing step (see Figure 4). A Poisseuille flow profile is imposed on the inflow ($x = -1$; $0 \leqslant y \leqslant 1$) and no slip conditions are imposed on the side walls. Neumann conditions are applied at the outflow that automatically sets the mean outflow pressure to zero.
3. For the 3D case, the Navier–Stokes equations are solved in a 3D backward facing step domain with parabolic inflow velocity profile, no slip condition on the side walls and Neumann conditions at the outflow.

We have used the finite element package SEPRAN [29] for our results in Section 5.1. The time reported in this section consists of construction time plus the time taken by an iterative solver.

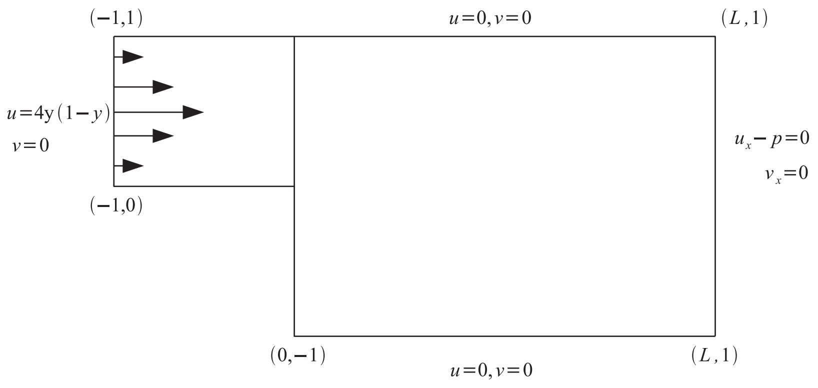

Figure 4. Backward facing step or L-shaped domain.

In Section 5.2 for comparison of various preconditioners, Matlab in combination with the IFISS package is used. For comparison of these preconditioners, we estimate the number of flops (floating point operations) used by the preconditioned method to approximate the solution. In the block triangular preconditioners that use MG as the inner solver, matrix–vector products and MG cycles consume the largest part of the flops. In the MG process, the number of flops consumed depends on the type of cycle involved. We use 1 $V(1,1)$ cycle in the preconditioning step and take as a slight underestimate that it is computationally just as expensive as one matrix vector product. The justification being that it involves one pre-smoothing, one post-smoothing, and one coarse grid correction. In all the experiments, the method is stopped when the iterative solver reaches the desired tolerance ($\|r_k\|_2/\|r_0\|_2 <$accuracy).

5.1. Properties of the saddle point ILU solver (SILU)

We have tested and compared the SILU preconditioner for the Stokes and Navier–Stokes problems in the domains mentioned above. In Table I Sloan renumbering for grid points is used to solve the Stokes problem in the 2D square domain for the $Q2$–$Q1$ discretization. For small problems, a direct method gives the solution faster than iterative methods coupled with the SILU preconditioner. However, memory requirements for the direct method are large compared with the iterative solution methods [30], and beyond a certain number of grid points the time required to solve the Stokes equations by a direct method increases considerably. The Krylov solvers converge faster for p-last per level than the p-last reordering of the unknowns. The time taken by the convergence of Bi-CGSTAB and GMRESR is less than GMRES(20). Also Bi-CGSTAB uses less matrix vector products than GMRES(20). The same convergence behavior has been found for the $Q2$–$P1$ discretization.

With the p-last per level reordering, the effect of renumbering the mesh by the Sloan or the Cuthill McKee algorithm is shown in Table II. The Sloan renumbering gives faster convergence than the Cuthill McKee renumbering for both $Q2$–$Q1$ and $Q2$–$P1$ discretizations. The difference in the number of iterations for both renumbering schemes is more pronounced in $Q2$–$P1$ discretization. The Sloan renumbering produces a much better profile than the Cuthill McKee renumbering. With a better profile, an incomplete LU decomposition gives a better approximation of the exact

Table I. Solution of the Stokes problem with the Q_2–Q_1 discretization in the square domain with an accuracy of 10^{-6}.

		16×16		32×32		64×64	
Solver	Renumber	Iterations	Time(s)	Iterations	Time(s)	Iterations	Time(s)
Direct	p-last	—	0.61	—	20.34	—	1378
	p-last per level	—	0.13	—	2.28	—	37
GMRES(20)	p-last	95	0.16	354	1.72	1800	44.0
	p-last per level	50	0.12	207	1.14	792	20.0
GMRESR	p-last	14	0.14	26	0.85	68	11.26
	p-last per level	12	0.12	21	0.72	43	7.27
Bi-CGSTAB	p-last	36	0.11	90	0.92	255	11.98
	p-last per level	25	0.09	59	0.66	135	6.74

Table II. Effect of mesh renumbering on convergence of Bi-CGSTAB for various discretizations in the backward facing step Stokes problem with p-last per level and accuracy$=10^{-6}$.

	$Q2$–$Q1$		$Q2$–$P1$	
Grid	Sloan iterations	Cuthill-McKee iterations	Sloan iterations	Cuthill-McKee iterations
8×24	9	15	29	97
16×48	22	32	40	288
32×96	59	65	73	1300

LU decomposition and increases the convergence of the preconditioned Krylov subspace method. However, if the problem is highly non-symmetric and far from being diagonally dominant, the norm of residual $R=A-\hat{L}\hat{U}$ is not a good estimate for the quality of the preconditioner. The Frobenius norm of the term $R(\hat{L}\hat{U})^{-1}=I-A(\hat{L}\hat{U})^{-1}$ gives better insight into the quality of the preconditioner. Even if R is small in norm, it may happen that the preconditioned matrix deviates largely from identity due to very large entries in $(\hat{L}\hat{U})^{-1}$[9].

In the case of the Navier–Stokes problem, the number of accumulated inner iterations increases with the increase in the Reynolds number and the number of grid points as shown in Figure 5. The linear equations in the inner iteration are solved with an accuracy of 10^{-2}. Figure 5 shows the accumulated number of inner iterations for solving the linear problems linearized by Picard and Newton. The SILU-preconditioned Krylov subspace methods with Sloan renumbering and p-last per level reordering were applied. There are two reasons for the increase in the accumulated inner iterations when the SILU preconditioner is employed:

With the increase in the Reynolds number, the number of outer iterations increases; however, the number of average inner iterations remains the same. Therefore, the convergence of the SILU preconditioner is hardly influenced by an increase in the Reynolds number.

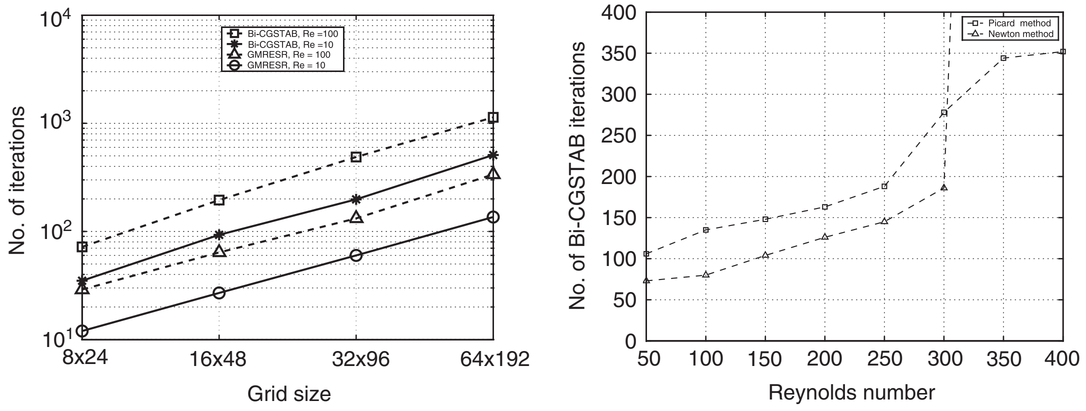

Figure 5. Effect of grid increase (left) and Reynolds number (right) on the inner iterations (accumulated) for the Navier–Stokes backward facing step problem with an 25 accuracy of 10^{-2} using the p-last per level reordering.

With the increase in the number of grid points, the number of inner iterations increases that gives rise to increase in the accumulated inner iterations. Change in the number of grid points has less effect on the outer iterations.

In most cases, it appears that the Newton method gives faster convergence results than the Picard method. However, for a high Reynolds number—due to a bad initial estimate—Newton's method diverges; hence, it is good practice to use only Picard iterations after the Stokes equations for high Reynolds number or a few Picard iterations followed by Newton iterations. Although the Picard method converges in more outer iterations, the average inner iteration required by the Picard method is less than those required by the Newton's method. Note that we did not use an upwind technique in our experiments. We expect that convergence will be better in combination with SUPG.

The Stokes and the Navier–Stokes equations are also solved for a 3D backward facing step. Results with $Q2$–$Q1$ elements for the Stokes problem are shown in Table III. The Cuthill McKee numbering shows faster convergence than the Sloan renumbering with both p-last and p-last per level reordering schemes. A Stokes problem is also solved with a direct solver in an $8\times8\times8$ grid with p-last per level ordering. It gives the solution in 20 s with the Sloan renumbering and 52 s with the Cuthill McKee renumbering. This approach consumes a large amount of memory. For the $Q2$–$P1$ elements, the Sloan renumbering gives faster convergence than the Cuthill McKee for both reordering methods. Results given in Table IV suggest that the Krylov subspace method converges in the same number of iterations with both orderings for the Cuthill McKee renumbering. Table V shows the results obtained from the solution of the Navier–Stokes problem. This reveals that the preconditioned Krylov methods have the same convergence behavior for various Reynolds numbers and grid sizes as we have observed in the Stokes problem.

The effect of grid stretching is shown in Table VI. We increased the number of elements in only one direction, so that the aspect ratio of the elements increases. Furthermore, some fill-in and lumping is introduced in the SILU preconditioner for a stretched grid. This improves the convergence of the iterative method. However, the CPU time and memory usage increase. Hence, it

Table III. Solution of the 3D Stokes backward facing step problem using $Q2$–$Q1$ elements with Bi-CGSTAB and accuracy $=10^{-4}$.

	Sloan				Cuthill McKee			
	p-last		p-last per level		p-last		p-last per level	
Grid	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)
$8\times8\times12$	41	5	39	4.88	37	3.76	27	3.26
$16\times16\times24$	137	82	138	122	125	83	66	46
$24\times24\times36$	341	693	325	725	265	566	123	275

Table IV. Solution of the 3D Stokes backward facing step problem using $Q2$–$P1$ elements with Bi-CGSTAB and accuracy $=10^{-4}$.

	Sloan				Cuthill McKee			
	p-last		p-last per level		p-last		p-last per level	
Grid	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)	Iterations	Time (s)
$8\times8\times12$	53	3.72	48	3.45	64	4.20	64	4.16
$16\times16\times24$	138	114	107	92	278	214	231	181
$24\times24\times36$	295	591	255	512	425	833	426	836

Table V. Accumulated inner iterations for the 3D Navier–Stokes backward facing step problem with p-last per level reordering.

Picard method	$Q2$–$Q1$		$Q2$–$P1$	
Grid	Sloan	Cuthill McKee	Sloan	Cuthill McKee
Reynolds number $=75$, *Bi-CGSTAB*, *accuracy* $=10^{-2}$				
$8\times8\times12$	147	76	253	350
$16\times16\times24$	575	222	598	934
$24\times24\times36$	8634	533	1994	2441
Reynolds number $=100$				
$8\times8\times12$	150	85	318	376
$16\times16\times24$	694	249	690	1432
$24\times24\times36$	—	576	1636	2668

suggests that fill-in and lumping should be used only when the ILU-preconditioned iterative method is diverging. We see a sharp dive (minimum) in the number of iterations for the 64×24 grid. For this behavior, we do not yet have an explanation. This has been observed for cells with an 8:3 ratio in the backward facing step. For both reordering schemes, the number of iterations increases with the increase in stretching, but there is a clear difference between Sloan and Cuthill McKee renumbering. Cuthill McKee sometimes diverges, while Sloan in combination with p-last per level always converges. The convergence is also improved in some cases where the incompressibility

Table VI. Solution of the Stokes problem in a stretched backward facing step with Bi-CGSTAB.

Grid	p-last iterations time (s)				p-last per level iterations time (s)			
$Q2$–$P1$	SILU	SILUF	Lumped	SILUF, Lumped	SILU	SILUF	Lumped	SILUF, Lumped
Sloan renumbering, *accuracy* $=10^{-4}$								
8×24	36 (0.04)	14 (0.06)	41 (0.04)	17 (0.06)	23 (0.03)	8 (0.02)	27 (0.04)	8 (0.03)
16×24	55 (0.15)	27 (0.2)	71 (0.18)	33 (0.24)	47 (0.12)	16 (0.11)	52 (0.14)	19 (0.12)
32×24	364 (1.78)	138 (1.36)	305 (1.5)	89 (1.13)	159 (0.8)	60 (0.63)	599 (2.91)	55 (0.61)
64×24	—	—	>3000*1	237 (5.31)*2	58 (0.65)	18 (0.45)	217 (2.22)	66 (1.26)
128×24	—	—	—	780 (34)*3	293 (6.1)*4	—	808 (16.34)	224 (7.6)
Cuthill McKee renumbering, *accuracy* $=10^{-4}$								
8×24	158 (0.16)	14 (0.08)	140 (0.15)	17 (0.1)	148 (0.15)	7 (0.07)	175 (0.17)	9 (0.07)
16×24	281 (0.64)	26 (0.34)	231 (0.51)	36 (0.41)	287 (0.65)	20 (0.25)	258 (0.58)	21 (0.26)
32×24	277 (1.36)	50 (0.96)	520 (2.53)	71 (1.24)	276 (1.31)	45 (0.88)	568 (2.71)	50 (0.96)
64×24	>3000	586 (18.59)	—	209 (7)*5	>3000	14 (0.88)	—	51 (2.1)
128×24	—	—	—	727 (47.34)*6	—	17 (2)	—	120 (9.33)

Table VII. Effect of ε on the convergence with cases labeled with $*$ in Table VI.

Case	ε	Iterations time (s)	Case	ε	Iterations time (s)
*1	1e−10	229 (2.37 s)	*2	1e−10	150 (3.52 s)
*3	1e−09	405 (17.8 s)	*4	1e−10	261 (5.46 s)
*5	1e−10	137 (4.73 s)	*6	1e−10	313 (21.0 s)

constraint is used, see Table VII. However, it is difficult to find a suitable value of ε. In Figure 6, we see that the reduction in number of iterations with the increase in the incompressibility constraint is not linear for the $Q2$–$P1$ discretization in the Stokes problem. Although the number of iterations decreases for higher values of ε, there is a large increase in the error norm as well. For the Navier–Stokes problem, the decrease in the number of iterations is not large as shown in Figure 7. With a slight compromise on the error norm, a suitable value of ε can be found in the range of 10^{-10}–10^{-6}. We have made the same observation for the $Q2$–$Q1$ elements.

5.2. *Comparison of the various preconditioners*

In this subsection, we consider only the 2D backward facing step problem. For this problem, we compare the SILU preconditioner (p-last per level with Sloan renumbering) with the preconditioners as discussed in Section 3. The preconditioners are tested for a stretched grid and varying Reynolds number. For the results in Table VIII, we have used the MG solver for solving these blocks (except for AL) for the Navier–Stokes equations linearized by Picard with the Reynolds number ranging from 100 to 400. In Figures 8 and 9, we have tested the preconditioner for the stretched grids in which we have used a direct solver for solving the $(1,1)$ and $(2,2)$ blocks of the LSC and AL preconditioners for the Navier–Stokes equations linearized by Newton's method. An approximation of the flops is computed where we assume that the number of iterations is the same for the direct solver and for the MG solver used for the inner iterations. We report here the number of iterations taken by an iterative solver in the final step of the non-linear iterations.

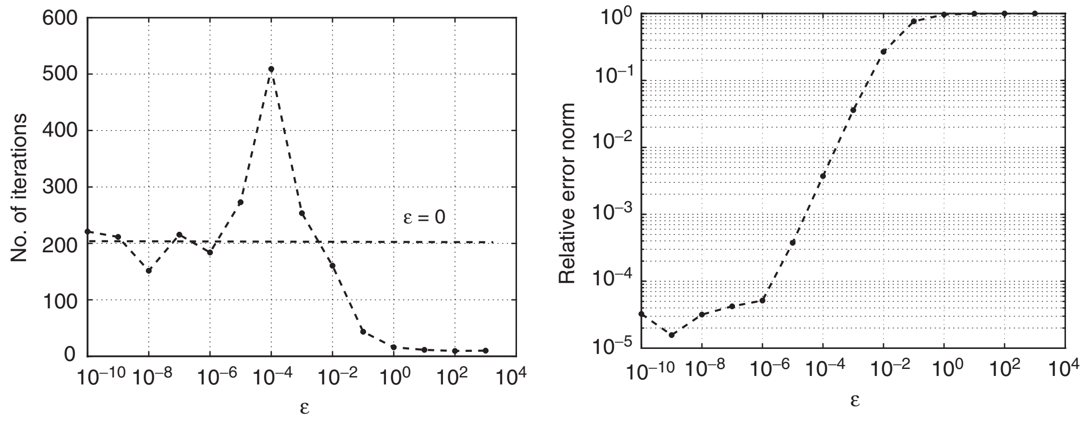

Figure 6. Effect of the incompressibility constraints (ε) on the number of iterations (left) and the relative error norm (right) in the backward facing Stokes problem using Bi-CGSTAB with an accuracy of 10^{-4}, and Cuthill McKee renumbering with p-last per level reordering for the 16×48 $Q2$–$P1$ discretization.

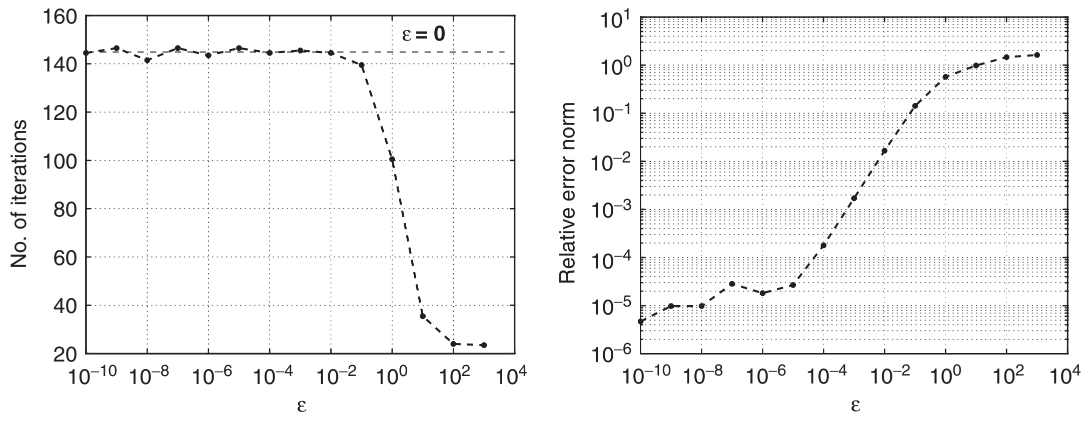

Figure 7. Effect of the incompressibility constraint (ε) on the number of iterations (left) and the relative error norm (right) in the backward facing Navier–Stokes problem using Bi-CGSTAB with an accuracy of 10^{-4}, and Cuthill McKee renumbering with p-last per level reordering for the 16×48 Q2-P1 discretization with $Re=100$ in the last step of the Picard iteration.

In Table VIII, we present our results by comparing the number of iterations and flops used in a preconditioned iterative solver. In terms of iterations, AL converges faster than all other preconditioners. However, looking at flops the SILU preconditioner is more efficient than the other preconditioners except for the grid size 64×192. On average, the number of iterations reduces to 20 for the 64×192 grid if extra fill-in is used in the SILU preconditioner. The number of flops in this case reduces to 180. SILU convergence is effected by an increase in the number of grid elements.

Table VIII. Comparison of the preconditioners using MG solver for $(1,1),(2,2)$ blocks of PCD and LSC preconditioners and direct solver for AL (flops estimate with MG solver for AL) with Bi-CGSTAB and accuracy $=10^{-4}$ using $Q2$–$Q1$ element discretization.

	PCD		SILU		LSC		AL($\gamma=1$)	
	Iterations	Mflops	Iterations	Mflops	Iterations	Mflops	Iterations	Mflops
$Re=100$								
8×24	40	3.7	9	0.6	24	4	4	1.5
16×48	36	15.3	13	3.9	19	14.9	5	10.7
32×96	39	70.9	21	27.5	13	44.4	4	39
64×192	61	458	55	297	13	185	4	169
$Re=200$								
8×24	77	7	18	1.2	48	8	4	1.5
16×48	75	31.7	14	4.3	36	28.4	4	8.5
32×96	63	114.6	22	28.8	25	85	6	58.8
64×192	89	668	48	259	17	241	5	211
$Re=400$								
16×48	165	69	17	5.2	62	48	4	8.5
32×96	146	264	24	30	38	129	4	39
64×192	114	855	45	242	32	453	4	169

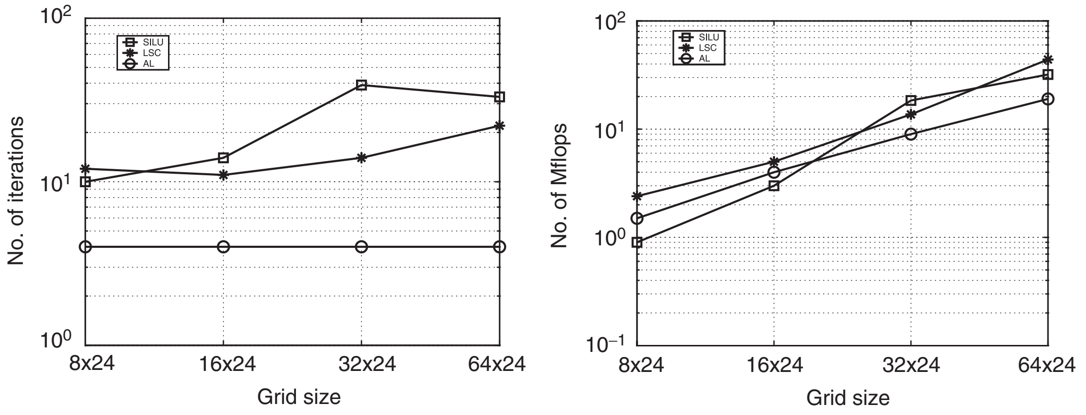

Figure 8. Comparison of the preconditioners using the $Q2$–$Q1$ element discretization in the backward facing step problem, with a stretched grid and Bi-CGSTAB with accuracy $=10^{-4}$ for $Re=100$, (left) iterations count, (right) Mflops count.

The preconditioners are also tested for stretched grids. The results are shown in Figures 8 and 9. On the basis of the results given in Table VIII, we exclude PCD. In both figures, the number of iterations taken by SILU is larger than those taken by LSC and AL. However, in flops, SILU is competing with LSC. AL seems to be more efficient and robust than the other preconditioners.

As we have seen, the number of iterations in the SILU preconditioner increases with an increase in the number of grid points and increases mildly with the increase in the Reynolds number.

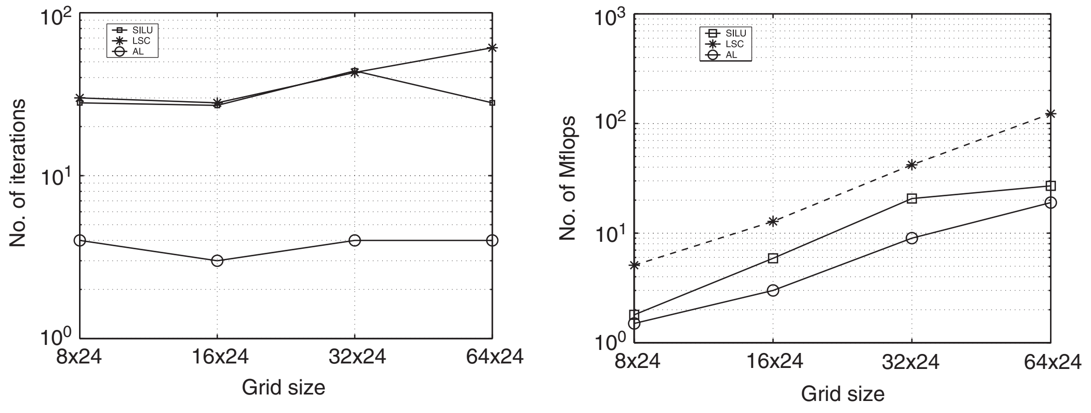

Figure 9. Comparison of the preconditioners using the $Q2$–$Q1$ element discretization in the backward facing step problem, with a stretched grid and Bi-CGSTAB with accuracy$=10^{-4}$ for $Re=200$, (left) iterations count, (right) Mflops count.

Therefore, for large problems the AL and LSC preconditioners may be more efficient due to their convergence independent of the Reynolds number and grid points. On the other hand, regarding the ease of implementation, requirements, and cost, the usefulness of SILU for a certain range of problems cannot be ignored.

6. CONCLUSIONS

In this paper, various preconditioners for the discretized Navier–Stokes equations have been compared. The classical ILU preconditioner is a popular method for the iterative solution of linear systems. It is well known that a straightforward application of this method can lead to breakdown or bad convergence due to small pivot elements. In order to prevent this, the grid points are first renumbered with the classical Cuthill McKee or the Sloan method. Thereafter, we reorder the unknowns such that on each level the velocity unknowns are ordered before the pressure unknowns. The resulting method is called the *saddle point ILU* (*SILU*) preconditioner. In our experiments we never observed any breakdown of the SILU decomposition and the convergence is fast for a large range of problems.

We observed the following properties of the SILU preconditioner:

- in 2D problems, the Sloan renumbering with p-last per level reordering leads to the best results for the Taylor–Hood and Crouzeix–Raviart elements;
- in 3D problems, Cuthill McKee renumbering gives fast convergence for the $Q2$–$Q1$ discretization, whereas for the $Q2$–$P1$ discretization, the Sloan renumbering gives a better convergence;
- GMRESR and Bi-CGSTAB are faster than GMRES(20);
- for non-linear iterations, it is better to use a few Picard iterations followed by Newton's iterations;
- the number of iterations increases with the increase in the number of elements in the grid and increases mildly with the increase in the Reynolds number.

We have compared the SILU preconditioner with some saddle point block preconditioners. The results show that the SILU preconditioner is simple to implement and competing with the saddle point block preconditioners. The efficiency of the block preconditioners depends highly on efficient solvers for the subproblems. Since these preconditioners are independent of the grid size and weakly dependent on the Reynolds number, their performance becomes better than the SILU preconditioner for large grid sizes and large Reynolds numbers.

Finally, increasing the grid stretching has a negative influence on the convergence behavior of the SILU preconditioner, whereas the block preconditioners depend only weakly on the stretching of the grid. Hence, the saddle point block preconditioners are attractive for high aspect ratios of the elements.

ACKNOWLEDGEMENTS

The author wishes to thank the Higher Education Commission of Pakistan and its Dutch handling agent Nuffic for making it possible to pursue a PhD study at the University of Technology in Delft, contract Ref: 2-17/PM-OS/Neth/2005.

REFERENCES

1. Benzi M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics* 2002; **182**(2):418–477.
2. Benzi M, Golub GH, Liesen J. Numerical solution of saddle point problems. *Acta Numerica* 2005; **14**:1–137.
3. Kay D, Loghin D, Wathen A. A preconditioner for the steady-state Navier–Stokes equations. *SIAM Journal on Scientific Computing* 2002; **24**(1):237–256.
4. Benzi M, Olshanskii MA. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing* 2006; **28**(6):2095–2113.
5. Elman HC, Silvester D, Wathen AJ. *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluids Dynamics*. Oxford University Press: Oxford, 2005.
6. Meijerink JA, Van Der Vorst HA. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Mathematics of Computation* 1977; **31**(137):148–162.
7. Duff IS, Meurant GA. The effect of ordering on preconditioned conjugate gradients. *BIT* 1989; **29**(4):635–657.
8. Dutto LC. The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier–Stokes equations. *International Journal for Numerical Methods in Engineering* 1993; **36**(3):457–497.
9. Benzi M, Szyld DB, Van Duin A. Orderings for incomplete factorization preconditioning of nonsymmetric problems. *SIAM Journal on Scientific Computing* 1999; **20**(5):1652–1670.
10. Brezzi F, Fortin M. *Mixed and Hybrid Finite Element Methods*. Springer: New York, 1991.
11. Dohrmann CR, Bochev PB. A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *International Journal for Numerical Methods in Fluids* 2004; **46**(2):183–201.
12. Fortin M. Old and new finite elements for incompressible flows. *International Journal for Numerical Methods in Fluids* 1981; **1**(4):347–364.
13. Dahl O, Wille SØ. 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. *International Journal for Numerical Methods in Fluids* 1992; **15**(5):525–544.
14. Wille SØ, Loula AFD. A priori pivoting in solving the Navier–Stokes equations. *Communications in Numerical Methods in Engineering* 2002; **18**(10):691–698.
15. Wille SØ, Staff O, Loula AFD. Efficient a priori pivoting schemes for a sparse direct Gaussian equation solver for the mixed finite element formulation of the Navier–Stokes equations. *Applied Mathematical Modelling* 2004; **28**(7):607–616.
16. Taylor C, Hood P. A numerical solution of the Navier–Stokes equations using the finite element techniques. *Computers and Fluids* 1973; **1**:73–100.
17. Crouzeix M, Raviart PA. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. *Rairo Analyse Numerique* 1973; **7**:33–76.

18. Cuvelier C, Segal A, Van Steenhoven AA. *Finite Element Methods and Navier–Stokes Equations*. Reidel Publishing Company: Dordrecht, Holland, 1986.
19. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1986; **7**(3):856–869.
20. Van Der Vorst HA. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. *SIAM Journal on Scientific and Statistical Computing* 1992; **13**(2):631–644.
21. Van Der Vorst HA, Vuik C. GMRESR: a family of nested GMRES methods. *Journal of Numerical Linear Algebra with Applications* 1994; **1**(4):369–386.
22. Murphy MF, Golub GH, Wathen AJ. A note on preconditioning for indefinite linear systems. *SIAM Journal on Scientific Computing* 2000; **21**(6):1969–1972.
23. Vuik C, Saghir A, Boerstoel GP. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *International Journal for Numerical Methods in Fluids* 2000; **33**(7):1027–1040.
24. Silvester D, Elman H, Kay D, Wathen A. Efficient preconditioning of the linearized Navier–Stokes equations for incompressible flow. *Journal of Computational and Applied Mathematics* 2001; **128**(1–2):261–279.
25. Elman H, Howle VE, Shadid J, Shuttleworth R, Tuminaro R. Block preconditioners based on approximate commutators. *SIAM Journal on Scientific Computing* 2006; **27**(5):1651–1668.
26. Benzi M, Choi H, Szyld D. Threshold ordering for preconditioning nonsymmetric problems. In *Proceedings of the Workshop on Scientific Computing*, Hong Kong, Golub G *et al.* (eds). Springer: Berlin, March 1997; 159–165.
27. Cuthill E, McKee J. Reducing the bandwidth of sparse symmetric matrices. *Proceedings of the 1969 24th National Conference*. ACM Press: 1969; 157–172.
28. Sloan SW. An algorithm for profile and wavefront reduction of sparse matrices. *International Journal for Numerical Methods in Engineering* 1986; **23**(2):239–251.
29. Segal G. *SEPRAN Introduction*. Ingenieursbureau Sepra: Leidschendam, NL, 1995.
30. Segal G, Vuik C. A simple iterative linear solver for the 3D incompressible Navier–Stokes equations discretized by the finite element method. *Technical Report DUT-TWI-95-64*, Delft, The Netherlands, 1995.