Preconditioners for the Steady Incompressible Navier-Stokes Problem

M. ur Rehman [*], C. Vuik [†] and G. Segal [‡]

Abstract—**In this paper we discuss preconditioners for the incompressible Navier-Stokes equations. In combination with Krylov subspace methods, they give a fast convergence for the solution of the Navier -Stokes equations. With the help of numerical experiments, we report some new findings regarding the convergence of these preconditioners. Besides that, a renumbering scheme for direct solvers and ILU preconditioners is introduced that improves the convergence of the solvers. We compare Bi-CGSTAB and a newly developed Krylov subspace method IDR(s) preconditioned with ILU. Both 2D and 3D experiments are used to measure the performance of the preconditioners.**

Keywords: block preconditioners, ILU preconditioners, Navier-Stokes, renumbering

1 Introduction

The incompressible Navier-Stokes equations, given as

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

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

are used to simulate fluid flow in a medium with the following properties: the fluid is incompressible and has a Newtonian character. Equation (1) represents the momentum equation and (2) is the continuity equation or mass conservation equation. ν is the viscosity (inversely proportional to the Reynolds number), $\mathbf{u}$ is the velocity vector and p is the pressure. For $\nu \to \infty$, the system of equations in (1) and (2) tends to a linear system of equations known as Stokes problem. The boundary value problem we consider, is system (1) and (2) posed on a two dimensional domain Ω, together with boundary conditions on $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ given by

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

where $\mathbf{w}$ is a given function.
The system given in (1) and (2) is discretized by the finite element method. Due to the presence of the convective term $(\mathbf{u}.\nabla\mathbf{u})$ in the momentum equation, the discretization of the Navier-Stokes equation leads to a system of non-linear equations. The Navier-Stokes system is linearized by Picard's method. In the Picard iteration method, the velocity from the previous iteration is substituted into the convective term. Starting with an initial guess $\mathbf{u}^{(0)}$ for the velocity field, Picard's iteration constructs a sequence of approximate solutions $(\mathbf{u}^{(k+1)}, p^{(k+1)})$ by solving a linear Oseen problem

$$-\nu\Delta\mathbf{u}^{(k+1)} + (\mathbf{u}^{(k)}.\nabla)\mathbf{u}^{(k+1)} + \nabla p^{(k+1)} = \mathbf{u} \;\; \text{in} \;\; \Omega, \tag{3}$$

$$\nabla.\mathbf{u}^{(k+1)} = 0 \;\; \text{in} \;\; \Omega, \tag{4}$$

in matrix notation

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{5}$$

$F = A + N$, where A is the viscous part, N is the contribution of convective term linearized by Picard's method, B^T is the gradient operator, and B is the divergence operator. The linearization of the Navier-Stokes problem gives rise to a saddle point problem, which means that there is a large block of zeros at the main diagonal.

Several techniques have been introduced to solve this system efficiently. Recently various preconditioners have been published, that can be used to accelerate the solution of system (5) by Krylov subspace methods [1–3]. We will discuss SIMPLE-type preconditioners as formulated by Vuik [3] in Section 2. Some remarks are also added on the importance of relaxation parameters in SIMPLE-type preconditioners.

[1]Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.050, Mekelweg 4, 2628 CD, Delft, The Netherlands; email: M.urRehman@tudelft.nl

[2]Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.060, Mekelweg 4, 2628 CD, Delft, The Netherlands; email: c.vuik@tudelft.nl

[3]Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.280, Mekelweg 4, 2628 CD, Delft, The Netherlands; email: a.segal@tudelft.nl

In Section 3, we discuss numerical solutions based on LU factorization (direct solver) and incomplete LU preconditioner. In the Navier-Stokes problem, LU/ILU factorization may fail due to zeros on the main diagonal unless partial pivoting is applied. 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. Various other orderings have been proposed in the literature [4–6], but our ordering scheme outperforms all of them. We also prove that our reordering scheme does not breakdown. Besides that, some features of a newly developed Krylov subspace method IDR(s) [7] are discussed and a comparison is made with Bi-CGSTAB preconditioned with our ILU.

In Section 4, numerical experiments are performed in 2D and 3D domains. We end with some conclusions in Section 5.

2 Preconditioners for the Navier-Stokes Equations

Preconditioning is a technique used to enhance the convergence of an iterative method to solve a large linear systems iteratively. Instead of solving a system $\mathcal{A}x = b$, one solves a system $P^{-1}\mathcal{A}x = P^{-1}b$, where P is the preconditioner. A good preconditioner should lead to fast convergence of the Krylov method. Furthermore, systems of the form $Pz = r$ should be easy to solve.

For the Navier-Stokes equations, the objective is to design a preconditioner, that increases the convergence of an iterative method independent of the Reynolds number and number of gridpoints. Secondly, the application of a preconditioner should be cheap. For more details, see [8]. We discuss here preconditioners for the incompressible Navier-Stokes equations.

2.1 SIMPLE(R) Preconditioner

SIMPLE (Semi Implicit Method for Pressure Linked Equations) [9], [10] is a classical algorithm for solving the Navier-Stokes equations, discretized by a finite volume technique. In this algorithm, to solve the momentum equations, the pressure is assumed to be known from the previous iteration. The newly obtained velocities do not satisfy the continuity equation since the pressure field is only a guess. Corrections to velocities and pressure are proposed to satisfy the discrete continuity equation. The SIMPLE algorithm can be derived from the block LU decomposition of the coefficient matrix (5)

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{6}$$

where $S = -BF^{-1}B^T$ known as the Schur complement matrix. The approximation $F^{-1} = D^{-1} = diag(F)^{-1}$ in (2,2) and (1,2) in L and U block matrices, respectively, leads to the SIMPLE algorithm. Define

$$\begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}. \tag{7}$$

Solve first

$$\begin{bmatrix} F & 0 \\ B & -BD^{-1}B^T \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{8}$$

and then u and p from (7). In the SIMPLE algorithm, the above two steps are performed recursively leading to:

SIMPLE algorithm:

1. Solve $Fu^* = r_u - B^T p^*$.
2. Solve $\hat{S}\delta p = r_p - Bu^*$.
3. update $u = u^* - D^{-1}B^T\delta p$.
4. update $p = p^* + \delta p$,

where pressure p^* is estimated from the prior iteration. D is the diagonal of the convection diffusion matrix F and $\hat{S} = -BD^{-1}B^T$ is an approximation of the Schur complement.

Vuik et al [3], used SIMPLE and its variants as a preconditioner to solve the incompressible Navier-Stokes problem. One iteration of the SIMPLE algorithm is used as a preconditioner with assumption $p^* = 0$. The preconditioner gives nice convergence if used in combination with the GCR method. However, the convergence decreases if the number of grid elements or Reynolds number increases. A variant of SIMPLE, SIMPLER gives convergence independent of Reynolds number. Instead of estimating the pressure p^* in the SIMPLE algorithm, p^* is obtained from solving a subsystem:

$$\hat{S}p^* = r_p - BD^{-1}((D - F)u^k + r_v), \tag{9}$$

where u^k is obtained from the prior iteration. In case SIMPLER is used as preconditioner, u^k is taken equal to zero. The classical SIMPLER algorithm proposed by

Patanker consists of two pressure solves and one velocity solve. However, in the literature the SIMPLER algorithm is formulated such that the steps of the algorithm are closely related to the Symmetric Block Gauss-Seidel method [3]. This form of the SIMPLER preconditioner can be written as:

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} u^k \\ p^k \end{pmatrix} + M_L^{-1} B_L \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^k \\ p^k \end{pmatrix} \right), \quad (10)$$

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix} \right), \quad (11)$$

where A represents the coefficient matrix given in (5), u^k and p^k in (10) are obtained from the previous step (both zero in our case) and

$$B_R = \begin{pmatrix} I & -D^{-1}B^T \\ 0 & I \end{pmatrix}, \; M_R = \begin{pmatrix} F & 0 \\ B & \hat{S} \end{pmatrix} \text{ and} \quad (12)$$

$$B_L = \begin{pmatrix} I & 0 \\ -BD^{-1} & I \end{pmatrix}, \; M_L = \begin{pmatrix} F & B^T \\ 0 & \hat{S} \end{pmatrix}. \quad (13)$$

The steps given in (10) and (11) contain two Poisson solves, two velocity subproblems solves- apposed to one velocity solve in the classical algorithm- and matrix vector updates. However, the extra velocity solve in formulation (10) and (11) has no significant effect on the convergence of the SIMPLER preconditioner. In the remainder of this paper, we will use SIMPLER with only one velocity solve.

SIMPLER is more expensive than SIMPLE. One iteration of the SIMPLER algorithm is approximately 1.3 times more expensive than the SIMPLE iteration [3]. SIMPLER convergence is also faster than the SIMPLE preconditioner. However, convergence with both preconditioners is decreased with an increase in the number of grid elements.

Effect of relaxation parameter

In the classical SIMPLE methods for finite volumes it is common practice to apply relaxation parameters to improve convergence. Unfortunately good choices for relaxation parameters can only be found by trial and error.

In our case we use SIMPLE as preconditioner, which means that we apply only one SIMPLE iteration per GCR step. In this case we introduce a relaxation parameter ω in the pressure part. The last step in the SIMPLE-type preconditioners is replaced by

$$p = p^* + \omega \delta p. \quad (14)$$

In contrast to the finite volume case, no relaxation parameter for the velocity part is used. The parameter ω is varied between 0 and 1. From our experiments it is clear that a proper choice of ω is more important when SIMPLE is used as iterative solver, than when it is used as preconditioner.

3 Reordering Scheme for LU/ILU Factorization

In this section we will discuss an a priori renumbering scheme to use both in the ILU preconditioner and a direct solver to solve the Navier-Stokes problem. From a practical point of view, it would be attractive, if standard classical iterative solution schemes, like 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 get a zero pivot, especially if velocities at some boundaries are prescribed and therefore both factorizations may fail. Pivoting, on the other hand, will result in a large increase of 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. We propose a new ordering that avoids breakdown of LU factorization. Our reordering schemes consist of two steps.

1. Renumbering of grid points, which can be accomplished by any renumbering method that gives an optimal profile. We use Cuthill McKee (CMK) [11] and Sloan [12] renumbering schemes for grid points.
2. The second step consisits of reordering of unknowns. Unknowns can be reordered as first all the velocity unknowns, followed by pressure unknowns in the grid. This is know as *p-last* ordering.

   A new type of reordering is introduced, in which the grid is divided into levels. Each level consists of a connected set of nodes. Thereafter, the unknowns are ordered per level. At each level, first velocity unknowns are placed and then followed by the pressure unknowns. We call it *p-last per level* reordering.

Let us define the notion of levels for Cuthill McKee. Suppose we have created levels 1 to i-1. Then level i is de-

fined 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. In the p-last per level reordering, one has to be careful at the start of this process. If, for example, the velocities in the first node, are prescribed, we start with a pressure unknown that gives rise to a zero pivot. Therefore, we always combine the first few levels, 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 combinations of 2 or 3 levels is sufficient. Note that the starting level has always a small contribution to the global profile [13].

3.1 Direct Solver

In a direct method, a matrix A can be written in the form

$$A = LU,$$

where L is a lower triangular matrix and U is an upper triangular matrix. We have to solve $LUx = b$, which can be done in the following steps
first solve $Ly = b$,
then solve $Ux = y$.

In a direct method, The LU factorization is the costly part of the computational process and the solution of the two steps is usually of minor cost. The elimination process fills the non-zero entries of a sparse matrix within a band or profile. So large numbers of entries have to be stored and the CPU time increases. Generally, the system arising from the discretization of the finite element method has a sparse structure, which means that it contains a large number of zeros. The aim of a sparse direct solvers is to avoid doing operations on zero entries and therefore to try to minimize the number of fill-ins. We may save the computational cost and CPU time with an efficient reordering strategy which can be used to modify the structure of the matrix.

In the Navier-Stokes problem, if, for a direct solver, we use the p-last ordering, 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 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 [5]. On the other hand, p-last per level, in combination with a suitable node renumbering strategy, produces a nearly optimal profile shown in Figure (1) and avoids the need for pivoting in case of direct solvers. It has been applied to many practical problems, without ever producing small pivots.

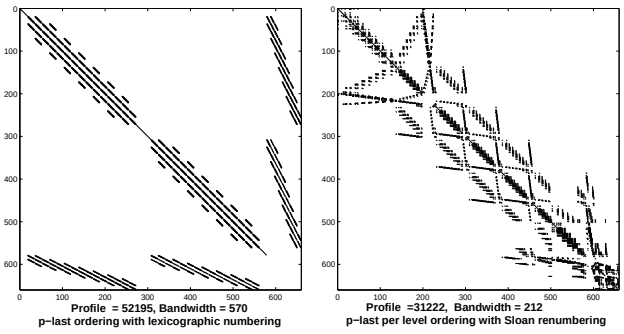

Figure 1: Effect of Sloan renumbering of grid points and p-last per level reordering of unknowns on the profile and bandwidth of the matrix

3.2 ILU Preconditioner

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. The sparseness structure is defined as follows:

$$(LD^{-1}U)_{i,j} \neq 0 \text{ for } (i,j) \in \mathcal{S},$$

where $\mathcal{S}$ consists of fill-in positions as the set of unknowns, that are directly connected. This implies that, zeros in the pressure block, are also part of the set $\mathcal{S}$, provided that there is a connectivity with velocity unknowns. The p-last reordering is used in SIMPLE preconditioner along with a renumbering scheme of grid points. This decreases the bandwidth of the coefficient matrix corresponding to the velocity and pressure unknowns. In general, the sparsity pattern of the coefficient matrix remains the same through out the non-linear iterations, therefore the reordering technique is applied once at the start of the iterations.

In our experiments, p-last per level in combination with a suitable renumbering for grid points is used. We have observed that *p-last per level* improves the convergence of the preconditioned iterative method and avoids the breakdown of ILU.

3.3 Breakdown of LU or ILU Factorization

Our strategy of p-last per level does not break down. The breakdown of ILU and LU due to p-last per level is only based on the choice of the first level. In many cases the first level contains prescribed boundary points. It might happen that our selected level gives rise to the pressure as a first row in the matrix, that in turn gives rise to a zero on the main diagonal. Therefore we kept our first level larger than the other levels. The question is, what should be the minimum number of points or nodes(unprescribed) in the first level so that our scheme encounter a danger of breakdown?

To explain how the minimum size of the first level must be chosen we consider a 2×2 Q2-Q1, Taylor-Hood element subdivision of a square shown in Figure (2). If all the velocities at the boundary are prescribed, restricting the initial set to the (oblique) dashed region, i.e. nodes 1 to 7, implies that in set 1 we have only 2 unknown velocities and 4 unknown pressures. Even if we start with the velocities, Gaussian elimination in these rows will not remove all zeros on the diagonal. This is the same reason why we have to satisfy the LBB condition. Adding node 8 to the dashed region makes the number of velocity unknowns in the first level equal to the number of pressure unknowns and the problem no longer exists.

So on the first level we need at least the same number of unprescribed velocity degrees of freedom as there are pressure degrees of freedom. Furthermore, the velocity unknowns should have a nonzero connection with the pressure unknowns.

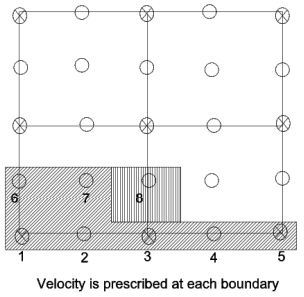

Figure 2: 2x2 Q2-Q1 grid

3.4 IDR(s) accelerated with ILU

In the past we used the Bi-CGSTAB [14] iterative method in combination with our ILU preconditioner. Theoretically Bi-CGSTAB gives the exact solution in $2N$ matrix-vector multiplications, with N the number of unknowns, provided exact arithmetic is used. A disadvantage of Bi-CGSTAB is its eratic convergence behavior.

Recently IDR(s)(Induced Dimension Reduction) [7] has been developed as a new Krylov alternative. The parameter s defines the size of a subspace of search vectors. The larger s, the more memory is required. IDR(1) has the same properties as Bi-CGSTAB. For $s > 1$ the method becomes more stable. The number of matrix-vector multiplications per iteration is equal to s, the number of iterations usually decreases for increasing s. Theoretically $N + N/s$ matrix-vector multiplications are necessary to get the exact solution. The reduction of the number of iterations for increasing s is not monotonic. Large values of s sometimes even do not improve performance of IDR(s) [15]. Usually s is taken in the order of 4. IDR(s) in combination with ILU is usually more stable than Bi-CGSTAB. In this paper we compare both methods.

4 Numerical Experiments

Numerical experiments are performed for the following benchmark problems:

1. Driven cavity problem; flow in a square cavity with enclosed boundary conditions and a lid moving from left to right given as:

$$y = 1; \ -1 \leq x \leq 1 | u_x = 1 - x^4,$$

   known as regularized cavity problem.

2. The L-shaped domain $(-1, L) \times (-1, 1)$, known as the backward facing step. A Poisseuille flow profile is imposed on the inflow ($x = -1$; $0 \leq y \leq 1$) and zero velocity conditions are imposed on the walls. Neumann conditions are applied at the outflow which automatically sets the mean outflow pressure to zero. Results are also performed in a 3D backward facing step.

The GCR method, [16] PCG [17], Bi-CGSTAB and IDR(s) are used in our experiments. Both direct solvers and ILU preconditioners are used to solve subsystems in the SIMPLE-type preconditioners. We divide the experiments into two sections; Section 4.1 which deals only with SIMPLE-type preconditioners and Section 4.3 which consists of a comparison of SIMPLE-type preconditioners with our ILU preconditioner. The iteration is stopped if the linear systems satisfy $\frac{\|r^k\|_2}{\|b\|_2} \leq tol$, where r^k is the residual at the kth step of the Krylov subspace method,

b is the right hand side, and *tol* is the desired tolerance value. Some abbreviations used are: *It.(s)* for number of iterations (time in seconds), *Mat.-Vec.* stands for matrix vector multiplications, *ts* for time in seconds and NC for no convergence. The accuracy of the inner solvers in the SIMPLER preconditioner is represented in the form $10^{p,u,p}$ (exponent for the pressure solve, velocity solve and pressure solve), while in SIMPLE, pressure is computed with accuracy 10^{-2} and the velocity 10^{-1} in the preconditioning steps. The grid size in the tables and figures refer to the number of Q2-Q1 elements. Numerical experiments are performed on an *Intel 2.66 GHz processor with 8GB RAM.*

4.1 SIMPLE-type Preconditioners

The Stokes problem is solved with SIMPLE and SIMPLER preconditioners using exact and inexact solvers for the subsystems shown in Table 1. For the inexact solution, we used preconditioned CG. If we use exact inner solves, SIMPLER converges faster than SIMPLE although SIMPLER requires an extra pressure solve. Both preconditioners seem efficient in CPU time if exact inner solves are used instead of inexact solves. However, for large problems in 2D and problems in 3D, exact inner solvers are not a cheap option to use. The convergence of the SIMPLE preconditioner is more effected by the increase in the grid size than the SIMPLER preconditioner. However, a positive aspect of the SIMPLE preconditioner we have observed is that the convergence of the SIMPLE preconditioner is independent of the accuracies used to solve the subsystems, while the SIMPLER preconditioner strongly depends on the inner accuracies. The larger the number of grid elements, the larger the accuracy requirement for the inner solver in the SIMPLER preconditioner. Iterations in the SIMPLER preconditioner can be reduced with the increase in inner accuracies. On the other hand, increasing inner accuracies will have no large effect on the convergence of the SIMPLE preconditioner.

The Navier-Stokes problem solved with varying Reynolds numbers is shown in Table 2. We report here the number of iterations taken by preconditioned solver after one Picard step. We see that SIMPLER is converging faster than SIMPLE. However, SIMPLER requires some suitable inner accuracy for convergence. From the table, it is clear that the inner accuracy problem arises only due to the increase in the number of grid elements. As the viscosity decreases, the number of iterations of both preconditioners increase. This increase is large in the SIMPLE preconditioner and mild in SIMPLER. Viscosity independent convergence with the SIMPLER preconditioner can be achieved only if subsystems are solved with a high accuracy.

Table 1: Solution of the Stokes cavity flow problem with preconditioned GCR(20) method with accuracy 10^{-6}.

Grid	SIMPLE		SIMPLER		
-	Exact	Inexact	Exact	Inexact	accuracy
	It. (ts)	It. (ts)	It. (ts)	It. (ts)	$10^{p,\ u,\ p}$
8×8	20(0.13)	25(0.19)	10(0.07)	14(0.14)	-2, -1, -2
16×16	37(1.84)	45(1.75)	15(0.89)	19(0.2)	-2, -1, -3
32×32	71(14.5)	89(24.8)	24(5.3)	40(12.6)	-2, -1, -3
64×64	121(132)	165(362)	40(47.5)	49(183)	-3, -2, -4

Table 2: The Navier-Stokes cavity flow problem with preconditioned GCR(20) method with accuracy 10^{-6}, subsystem in the preconditioners are solved inexactly with ILU preconditioned Bi-CGSTAB.

ν	SIMPLE	SIMPLER	
	It. (ts)	It. (ts)	$10^{p,\ u,\ p}$
0.02	129(42)	37(27)	-3, -2, -3
0.01	179(68)	38(31)	-3, -2, -3
0.002	677(245)	70(66)	-3, -2, -3
0.001	1086(431)	118(96)	-3, -2, -3

Effect of relaxation

Experiments revealed that application of a relaxation parameter for the velocity gives no improvement in the convergence. Therefore, we restrict ourselves to varying the relaxation parameter ω in (14). Since relaxation did not improve the convergence of SIMPLER only SIMPLE is considered.

In the case of Stokes, choosing ω properly, might reduce the number of iterations with a factor 3 or 4. For example, in Figure (3), the optimal value of ω (0.05) reduces the number of outer iterations from 193 to 59 for a 64×64 grid. The reduction in the number of inner iterations - not shown in figure - is from 1400 to 77 for the velocity subsystem and 3600 to 1200 for the pressure subsystem.

Table 3 shows the effect of ω for various values of ν in the Navier-Stokes equations. We can see from the table that a proper value of ω might give some gain, but the profit is only small compared to that of the Stokes problem. Furthermore it is clear that the optimal value of ω is different for Stokes and Navier-Stokes. the probable

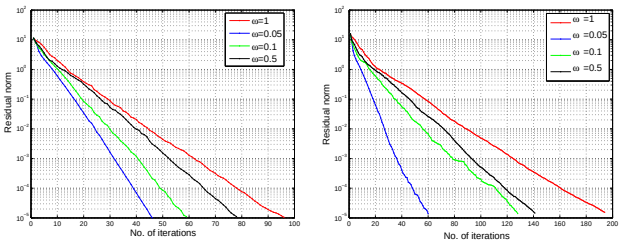

Figure 3: The Stokes problem solved in Q2-Q1 discretized driven cavity problem with varying ω: 32×32 grid (Left), 64×64 grid (Right).

cause is that the pressure approximation in the second step of the non-linear iteration is much better than in first iteration.

Table 3: Effect of relaxation on the Navier-Stokes problem with a solution accuracy 10^{-6}.

ν	$\omega = 1$	$\omega = 0.5$	$\omega = 0.35$	$\omega = 0.3$	$\omega = 0.2$
	It.	It.(s)	It.	It.	It.
0.02	657	641	552	552	563
0.01	870	803	773	857	783
0.001	7637	7080	6800	6666	NC

4.2 Impact of reordering on the direct solver

In this section, we present some results to see how our reordering strategy effects the efficiency of the direct solver. We report our findings with our renumbering scheme. Our renumbering scheme effectively reduces the profile and bandwidth of the matrix. In Table 4, we see the reduction with Sloan and Cuthill McKee renumbering method with p-last per level reordering of unknowns. Profile and bandwidth reduction is computed by dividing profile and bandwith with p-last by p-last per level. Profile reduction with the Sloan method is better than Cuthill McKee, while in bandwidth reduction Cuthill McKee performs better than Sloan. Thus, our reordering method reduces the memory and work and computation time if the system is solved with a direct solver.

To prove numerically that our reordering scheme improves the efficiency of the direct solver, the Stokes problem is solved with a direct solver with various renumbering and reordering combinations shown in Table 5. With p-last reordering, - with various renumbering schemes - we do not see much difference in CPU time consumed by the direct solver to get the exact solution. However, using p-last per level, the efficiency of the direct solver increases enormously. Sloan renumbering with p-last per level reordering gives better results than the other combinations. Though a better choice of a renumbering scheme also enhances the efficiency of the direct solver, we see that the increase is largely due to the p-last per level reordering strategy.

Table 4: Profile and bandwidth reduction in the backward facing step with Q2-Q1 discretization.

Grid	Profile reduction		Bandwidth reduction	
-	Sloan	CMK	Sloan	CMK
4×12	0.37	0.61	0.18	0.17
8×24	0.28	0.54	0.13	0.08
16×48	0.26	0.5	0.11	0.04
32×96	0.25	0.48	0.06	0.02

Table 5: The Stokes backward facing step solved with direct solver with Q2-Q1 discretization.

Grid	p-last		
-	Lexicographic	CMK	Sloan
16×48	5.6s	3.9s	3.1s
24×72	44.3s	33.4s	28s
32×96	205s	160s	142s
Grid	p-last per level		
16×48	3.15s	0.25s	0.13s
24×72	21s	1.14s	0.54s
32×96	88s	3.3s	1.5s

4.3 Comparison: ILU Preconditioner and SIMPLE-type Preconditioner

The renumbering of grid points and reordering of unknowns is used in the ILU preconditioner to solve the Stokes and the Navier-Stokes problem. In Figure (4), we see that

- p-last per level with Sloan and CMK give aster convergence than p-last for the 2D backward facing step Stokes problem,
- p-last per level with Sloan renumbering is faster than p-last per level with CMK renumbering,
- the number of iterations increases with the increase in the number of grid elements.

In the onward experiments, p-last per level reordering of unknowns will be used in combination with the Sloan and CMK renumbering schemes.

The 3D Stokes and the Navier-Stokes backward facing step problem are solved with the preconditioners discussed in this paper. Results given in Table 6 and 7 reveal that our renumbering method performs better than the block preconditioners. In 2D the ILU preconditioner with the Sloan renumbering is performing faster than ILU computed with CMK, however in 3D, it is the other way around. ILU with CMK renumbering gives better convergence than the Sloan renumbering. The SIMPLER preconditioner seems not to be applicable without accurate inner solvers which makes SIMPLER an expensive option to use as preconditioner. To achieve convergence for SIMPLER, for the two finest grids in Table 6 and 7, it was necessary to use an inner accuracy of more than 10^{-4} to solve inner subsystems. On the other hand SIMPLE shows robust convergence behavior with approximate inner solves. A common aspect of all these preconditioners is that convergence with these preconditioner is dependent on the grid size.

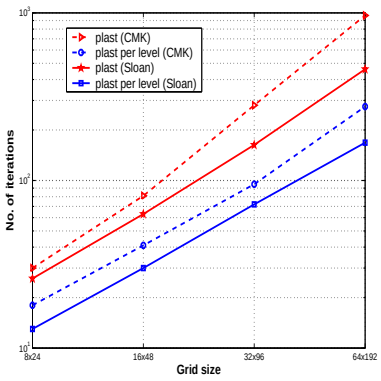

Figure 4: The 2D Stokes backward facing step problem solved with ILU preconditioned Bi-CGSTAB method with accuracy 10^{-6}.

Table 6: Solution of the 3D Stokes backward facing step with accuracy 10^{-6}.

Grid	GCR(20)		Bi-CGSTAB	
	SIMPLE	SIMPLER	CMK	Sloan
	It.(ts)	It.(ts)	It.(ts)	It.(ts)
$8 \times 8 \times 24$	31(4.2)	28(6)	24(1.28)	33(1.7)
$16 \times 16 \times 48$	74(142)	60(460)	54(26)	70(33)
$24 \times 24 \times 72$	104(1089)	NC	99(165)	122(200)

Table 7: Solution of the 3D Navier-Stokes backward facing step ($16 \times 16 \times 48$) with preconditioned GCR(20) with accuracy 10^{-2} and 10^{-4} in the Picard linearization. The accumulated number of iterations are reported.

	SIMPLE	SIMPLER	CMK	Sloan	Picard
	It.(ts)	It.(ts)	It.(ts)	It.(ts)	It.
0.02	300(789)	92(869)	225(120)	271(159)	7
0.01	464(1150)	115(925)	311(159)	368(200)	9
0.004	773(1448)	155(919)	856(317)	649(293)	12

4.4 Experiments with IDR(s)

In this section, we compare Bi-CGSTAB and IDR(s) preconditioned with ILU. In Figure (5), the number of iterations and CPU-time for the solution of the Stokes backward facing step problem for two different grid are plotted. We see that an increase of s from 1 to 2 reduces the CPU-time considerably (especially for the fine grid), but further increase of s has no significant profit. The increase of s does not give a monotone decrease of iterations.

In Table 8 we see the number of matrix-vector multiplications and CPU-time for the Stokes driven cavity problem solved on a uniform grid. Since 1 iteration of Bi-CGSTAB costs 2 matrix-vector multiplications and IDR(s) requires s multiplications per step this is the best way of comparison. The difference between Bi-CGSTAB and IDR(4) is not very significant. However, the table suggests that this difference increases for increasing grid size.

The main reason to use IDR is the better performance in case the ILU-preconditioner is not so efficient. To show this we consider a 2D lid driven cavity with a grid refined in the region where we have strong gradients. The subdivision is symmetric with respect to midpoints of the square (Figure (6)). The stretch factor SF is defined as the ratio of the largest and smallest edge in the grid. Table 9 shows the effect of the stretch factor for the Stokes problem on a 128×128 grid. The optimal choice, IDR(7), shows a much better performance for increasing SF than Bi-CGSTAB.

In Table 10, the backward facing step Stokes problem is solved on stretched grid in which the number of elements perpendicular to the flow direction are increased gradually. Bi-CGSTAB shows convergence only for the first case while IDR(s) converges for all.

In general, IDR(s) proved to be more stable than Bi-

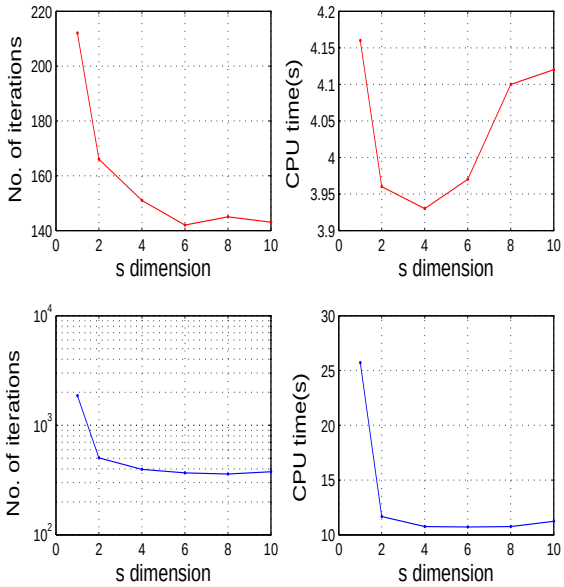

Figure 5: The 2D Stokes backward facing step problem solved with ILU preconditioned IDR(s) method with varying s dimension: 32×96 grid (Top), 64×96 grid (Bottom).

CGSTAB. For the problems, where Bi-CGSTAB shows convergence, IDR(s) always shows convergence. However, in some cases, we observed divergence of Bi-CGSTAB whereas IDR(s) converged.

5 Conclusions

In this paper, various preconditioners for the discretized Navier-Stokes equations have been compared. SIMPLE - type and ILU preconditioners in combination with special renumbering scheme are discussed in this paper.

- 2D experiments show that SIMPLER performs better than SIMPLE. However, the convergence of the SIMPLER preconditioner strongly depends on the accuracies of the subsystem solvers. Increase in problem size, hardens the demand to use an accurate inner solver. This limits the use of the SIMPLER preconditioner in 3D. On the other hand, though SIMPLE converges in more outer iterations than SIMPLER, it does not require an increase of accuracy of the inner subsystem. This makes the SIMPLE preconditioner suitable for a wide range of problems. The viscosity independent convergence of

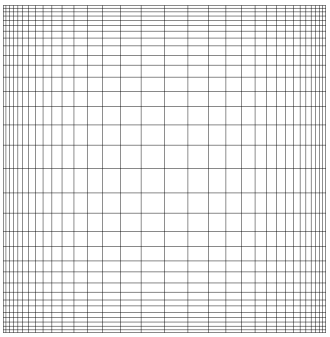

Figure 6: A 32×32 grid with stretch factor = 8.

Table 8: ILU preconditioned Krylov subspace methods comparison with increasing grid size for the driven cavity Stokes flow problem.

Grid	Bi-CGSTAB	IDR(4)
	Mat.-Vec. (ts)	Mat.-Vec. (ts)
16×16	38(0.01)	33(0.01)
32×32	90(0.14)	75(0.14)
64×64	214(1.6)	159(1.4)
128×128	512(16)	404(15)
256×256	1386(183)	1032(156)

Table 9: ILU preconditioned Krylov subspace methods comparison with increasing stretch factor for the driven cavity Stokes flow problem.

Stretch factor	Bi-CGSTAB	IDR(7)
	Mat.-Vec.(ts)	Mat.-Vec.(ts)
1	512(16)	370(15)
4	804(25)	432(17.5)
7	1104(34)	524(21)
11	1354(43)	612(24)
15	1576(49)	663(26)

Table 10: ILU preconditioned Krylov subspace methods comparison for the backward facing step Stokes problem.

Grid	Bi-CGSTAB	IDR(s)	
	Mat.-Vec.(ts)	Mat.-Vec.(ts)	s
32×96	214(1.3)	168(1.26)	4
64×96	NC	597(7.7)	4
96×96	NC	933(18)	4
128×96	NC	1105(31)	8

SIMPLER can only be achieved with exact solves for the subsystem. Proper choice of the relaxation parameter can reduce the number of outer/inner iterations if Stokes problem is solved with the SIMPLE preconditioner.

- Our special reordering scheme has first been developed for direct solvers. Results show that our reordering scheme reduces both memory and CPU time usage in a direct solver.

- For the ILU preconditioner, the p-last per level reordering scheme gives better convergence than p-last. p-last per level reordering reduces the profile and bandwith of the matrix and avoids breakdown of LU/ILU. In 2D, with p-last per level reordering, Sloan performs better than CMK. In 3D, CMK gives faster convergence than Sloan. The convergence of all these preconditioners strongly depends on the grid size. Compared to the other preconditioner discussed, SIMPLE convergence is more effected by the decrease in viscosity. Besides simple and cheaper implementation, our ILU preconditioner performs better than SIMPLE-type preconditioners.

- Our ILU preconditioner has been combined with both Bi-CGSTAB and IDR(s). It is shown that IDR(s) is more stable especially for increasing grid size and stretched meshes.

References

[1] M. Benzi and M. A. Olshanskii. An Augmented Lagrangian-Based Approach to the Oseen Problem. *SIAM J. Sci. Comput.*, 28(6):2095–2113, 2006.

[2] H. Elman, V. E. Howle, J. Shadid, R . Shuttleworth, and R. Tuminaro. Block Preconditioners Based on Approximate Commutators. *SIAM J. Sci. Comput.*, 27(5):1651–1668, 2006.

[3] C. Vuik, A. Saghir, and G. P. Boerstoel. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *Int. J. Numer. Meth. Fluids*, 33(7):1027–1040, 2000.

[4] O. Dahl and S. Ø. Wille. 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. *Int. J. Numer. Meth. Fluids*, 15(5):525–544, 1992.

[5] S. Ø. Wille and A. F. D. Loula. A priori pivoting in solving the Navier-Stokes equations. *Commun. Numer. Meth. Engng.*, 18(10):691–698, 2002.

[6] S. Ø. Wille, O. Staff, and A. F. D. Loula. Efficient a priori pivoting schemes for a sparse direct Gaussian equation solver for the mixed finite element formulation of the Navier-Stokes equations. *Appl. Math. Modelling*, 28(7):607–616, July 2004.

[7] Peter Sonneveld and Martin B. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems. *SIAM J. Sci. Comput.*, To appear.

[8] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.

[9] P. Wesseling. *Principles of computational fluid dynamics*, volume 29. Springer Series in Computational Mathematics, Springer, Heidelberg, 2001.

[10] S. V. Patankar. *Numerical heat transfer and fluid flow.* McGraw-Hill, New York, 1980.

[11] E. Cuthill and J. McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172. ACM Press, 1969.

[12] S. W. Sloan. An algorithm for profile and wavefront reduction of sparse matrices. *Int. J. Numer. Meth. Engng.*, 23(2):239–251, 1986.

[13] M. ur Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. *Int. J. Num. Meth. in Fluids*, 57:1731–1751, 2008.

[14] 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. Comput.*, 13(2):631–644, 1992.

[15] Martin B. van Gijzen and Peter Sonneveld. The IDR(s) method for solving nonsymmetric systems: a performance study for CFD problems. *To appear in: Proceedings of RIMS 2007*, 2007.

[16] C. Eisenstat, H. C. Elman, , and M. H. Schultz. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM J. Numer. Anal.*, 20(2):345–357, April 1983.

[17] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–435, 1952.