INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2009; **61**:432–452
Published online 20 November 2008 in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/fld.1957

SIMPLE-type preconditioners for the Oseen problem

M. ur Rehman[1,*,†], C. Vuik[2] and G. Segal[3]

[1]*Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.050, Mekelweg 4, 2628 CD, Delft, The Netherlands*
[2]*Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.060, Mekelweg 4, 2628 CD, Delft, The Netherlands*
[3]*Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.280, Mekelweg 4, 2628 CD, Delft, The Netherlands*

SUMMARY

In this paper, the steady incompressible Navier–Stokes equations are discretized by the finite element method. The resulting systems of equations are solved by preconditioned Krylov subspace methods. Some new preconditioning strategies, both algebraic and problem dependent are discussed. We emphasize on the approximation of the Schur complement as used in semi implicit method for pressure-linked equations-type preconditioners. In the usual formulation, the Schur complement matrix and updates use scaling with the diagonal of the convection–diffusion matrix. We propose a variant of the SIMPLER preconditioner. Instead of using the diagonal of the convection–diffusion matrix, we scale the Schur complement and updates with the diagonal of the velocity mass matrix. This variant is called modified SIMPLER (MSIMPLER). With the new approximation, we observe a drastic improvement in convergence for large problems. MSIMPLER shows better convergence than the well-known least-squares commutator preconditioner which is also based on the diagonal of the velocity mass matrix. Copyright © 2008 John Wiley & Sons, Ltd.

Received 30 June 2008; Revised 2 October 2008; Accepted 2 October 2008

KEY WORDS: Navier–Stokes equations; finite element method; block preconditioners; SIMPLE-type schemes; iterative methods; incompressible fluids

1. INTRODUCTION

Numerical solution of the incompressible Navier–Stokes equations is an active area of scientific research nowadays. Solving the resulting linear system efficiently is of primary interest, because most of the CPU time and memory is consumed in solving the linear systems. The steady

*Correspondence to: M. ur Rehman, Delft University of Technology, Faculty EEMCS, Delft Institute of Applied Mathematics, 07.050, Mekelweg 4, 2628 CD, Delft, The Netherlands.
†E-mail: M.urRehman@tudelft.nl

incompressible Navier–Stokes equations are given as

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

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

Equations (1) and (2) are known as the momentum equations and the continuity equation, respectively. $\mathbf{u}$ is the velocity vector, p is the pressure and ν is the viscosity that is inversely proportional to the Reynolds number. Ω is a 2 or 3D domain with a piecewise smooth boundary $\partial\Omega$ with boundary conditions on $\partial\Omega=\partial\Omega_E\cup\partial\Omega_N$ given by

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

Discretization of (1) and (2) by the finite element method (FEM) leads to a nonlinear system. After linearization by Picard, or Newton, the linear system can be written as

$$\begin{bmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix}=\begin{bmatrix} f \\ g \end{bmatrix} \tag{3}$$

where $F\in\mathbb{R}^{n\times n}$ is a convection–diffusion operator, $B\in\mathbb{R}^{m\times n}$ is a divergence operator and $m\leqslant n$. n is the number of velocity unknowns and m is the number of pressure unknowns. The system is sparse, symmetric indefinite in the case of the Stokes problem and unsymmetric indefinite in the Navier–Stokes problem. The system (3) is obtained from a finite element discretization that satisfies the LBB condition. In case where the LBB condition is not satisfied, we need some stabilization scheme in the continuity equation. In that case the right-under block in the matrix is no longer zero.

To solve the linear system (3), Krylov subspace methods with some suitable preconditioning techniques are used. For most applications, convergence of Krylov subspace methods depends on the spectrum of the coefficient matrix. A preconditioner P transforms the linear system $Ax=b$ to a preconditioned system $P^{-1}Ax=P^{-1}b$, such that $P^{-1}y$ should be cheap to compute, and $P^{-1}A$ must have a favorable spectrum for convergence. In general, preconditioning techniques based on algebraic and physics-based approaches are widely used. Algebraic type preconditioners are based on an ILU factorization or an approximate inverse of the coefficient matrix, where some pivoting or *a priori* reordering strategies are used that makes the preconditioner stable and effective [1–9]. The main properties of these type of preconditioners are:

- extra knowledge about the system is not needed,
- cheap and simple implementation.

Algebraic preconditioners applied to the complete system (3) may breakdown due to zero pivots. This problem can be solved by pivoting, which is in general very expensive. An alternative is to apply a suitable *a priori* renumbering. In Section 2, we will shortly discuss the saddle point ILU (SILU) preconditioner, which is based on this strategy.

An alternative approach is the use of block preconditioners based on block factorization of the coefficient matrix. Separate subsystems for velocity and pressure are solved during each iteration. An important aspect of this approach is a good approximation of the Schur complement. Examples of such preconditioners can be found in [10–19]. The final goal is to develop preconditioners that give convergence independent of the Reynolds number. Dependence of the number of nodal

DOI: 10.1002/fld

points should only be visible in the inner solves. In Section 3, we treat some block triangular preconditioners based on an approximation of the Schur complement by least-squares commutator (LSC). An overview of preconditioners for saddle point problems is given in [20, 21].

In Section 3, we will also discuss the semi implicit method for pressure-linked equations-type preconditioners. SIMPLE [22, 23] is a classical algorithm for solving the Navier–Stokes equations, discretized by a finite volume technique. SIMPLE and variants of SIMPLE (SIMPLER, SIMPLEC) are also used as preconditioners [17] or smoother in multigrid. These preconditioners also belong to the block preconditioners category. The convergence of the SIMPLER preconditioner is considered to be independent of the Reynolds number. However, an increase in the number of grid elements effects the convergence of SIMPLER.

In this paper, we will discuss the effect of scaling on SIMPLER and LSC. We will also discuss the relationship between LSC and SIMPLE pointed out by Elman *et al.* [12, 24]. In the SIMPLER preconditioner, we use some modification in the approximation of the Schur complement that makes it more efficient to use for a wide range of grids; we call it as modified SIMPLER (MSIMPLER). The comparison with the LSC and SILU shows that the MSIMPLER convergence is better than LSC and SILU. Moreover, the preconditioner is cheaper than SIMPLER and LSC. In Section 4, some numerical experiments are presented for two benchmark problems; the backward facing step and the lid-driven cavity flow. In Section 5 we end with our conclusions.

2. THE SADDLE POINT ILU PRECONDITIONER (SILU)

The saddle point ILU preconditioner is based on an incomplete factorization of the complete coefficient matrix with an *a priori* renumbering that makes the preconditioner applicable to saddle point problems [9]. Two kinds of reordering are introduced for this preconditioner:

1. Renumbering of grid points, which can be accomplished by any renumbering method that gives an optimal profile. Examples are the techniques described by Sloan [25] and Cuthill McKee [26].
2. Since we are dealing with saddle point problems, zero pivots may arise during ILU decomposition. An obvious way to avoid this problem is to renumber the unknowns in the sequence: first all the velocity unknowns and then the pressure unknowns. We call this *p-last* ordering. A more sophisticated reordering of unknowns is the so-called *p-last per level* reordering. The grid is subdivided into levels, where each level is a connected set of nodes. Thereafter unknowns are reordered per level, first the velocity unknowns and then the pressures.

After renumbering, ILU decomposition is applied to the reordered coefficient matrix A. The sparseness structure is defined as follows:

$$(LD^{-1}U)_{i,j} \neq 0 \quad \text{for } (i, j) \in \mathcal{S} \tag{4}$$

where $\mathcal{S}$ consists of those entries of A that are filled by the standard finite element assembly procedure. Thus, some elements in the pressure zero block are still part of $\mathcal{S}$ although their corresponding matrix coefficients are zero.

Both p-last and p-last per level avoid breakdown of the ILU preconditioner. The profile with the p-last ordering is relatively large compared with that of the p-last per level. Owing to the local block reordering, zero pivots become non-zero, during factorization, and no *a posteriori* pivoting is required.

DOI: 10.1002/fld

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 one, and if necessary this process is repeated. In practice, a combination of two or three levels is sufficient. Note that the starting level has always a small contribution to the global profile [9]. In our experiments, p-last per level in combination with a suitable renumbering of 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. The convergence of the SILU preconditioner, however, depends on the size of the grid and depends mildly on the Reynolds number.

3. BLOCK PRECONDITIONERS FOR THE INCOMPRESSIBLE NAVIER–STOKES PROBLEM

Block preconditioners are based on a block factorization of the incompressible Navier–Stokes matrix (3). They are either based on a block LDU factorization of (3)

$$\begin{bmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{bmatrix} = LDU = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^{\mathrm{T}} \\ 0 & I \end{bmatrix} \tag{5}$$

where $S = -BF^{-1}B^{\mathrm{T}}$ is known as the Schur complement matrix, or on the block DU formulation:

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

Preconditioners based on formulation (6) are known as block triangular preconditioners (P_t). The most expensive part of the block preconditioner is the inverse of F and S. The most important part of block preconditioners consists of cheap but suitable approximations of F^{-1} and S^{-1}. $F^{-1}x = y$ is computed by solving $Fy = x$ approximately, and in the same way $S^{-1}x = y$ by approximating $Sy = x$.

In general, block preconditioners consist of some good and cheap approximations to F^{-1} and S^{-1} along with matrix vectors multiplications and updates. F^{-1} is solved approximately, whereas S^{-1} is first approximated and then solved.

In order to investigate the spectral properties of the preconditioned matrix, one can consider the following generalized eigenvalue problem:

$$\begin{bmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & B^{\mathrm{T}} \\ 0 & S \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} \tag{7}$$

This eigenvalue problem has eigenvalues $\lambda = 1$ of multiplicity n, and the remaining eigenvalues depend on approximation to the Schur complement

$$BF^{-1}B^{\mathrm{T}}p = \mu_i Sp$$

where μ_i are the eigenvalues corresponding to the Schur complement matrix [24]. From the eigenvalues, it is evident that the convergence with the preconditioners P_t strongly depends on

DOI: 10.1002/fld

the approximation to the Schur complement matrix. The better the approximation to the Schur complement the faster the convergence with P_t. Some preconditioners with nice convergence properties are published in [11, 12, 14].

Based on the experiments published in [9], we decided to restrict ourselves to the LSC preconditioner because that appeared to be the best converging one of the block preconditioners investigated in this paper. This preconditioner is shortly recapitulated in the next section.

3.1. Block preconditioners based on approximate commutators

Based on the idea that the commutator of the convection–diffusion operator on the velocity space, multiplied by the gradient operator, with the gradient operator acting on the convection–diffusion operator on pressure space is small, Kay *et al.* [14] introduced an approximation to the Schur complement

$$\varepsilon_h = (Q_u^{-1}F)(Q_u^{-1}B^{\mathrm{T}}) - (Q_u^{-1}B^{\mathrm{T}})(Q_p^{-1}F_p) \tag{8}$$

where Q_u, the velocity mass matrix, and Q_p, the pressure mass matrix, are scaling matrices. F_p is a discrete convection–diffusion operator on pressure space. The multiplication by Q_u^{-1} and Q_p^{-1} transforms quantities from integrated values to the nodal values. Pre-multiplication of (8) by $BF^{-1}Q_u$, post-multiplication by $F_p^{-1}Q_p$ and assuming that the commutator is small, leads to the Schur approximation

$$BF^{-1}B^{\mathrm{T}} \approx BQ^{-1}B^{\mathrm{T}}F_p^{-1}Q_p \tag{9}$$

The approximation given in (9) is named as pressure convection–diffusion (PCD) preconditioner, in which the expensive part $BQ_u^{-1}B^{\mathrm{T}}$ in (9) is also replaced by its spectral equivalent matrix A_p known as the pressure Laplacian matrix. The updated form of PCD is given by

$$S = -BF^{-1}B^{\mathrm{T}} \approx -A_pF_p^{-1}Q_p \tag{10}$$

Instead of building two extra operators F_p and A_p in (10), Elman *et al*. derived a relation for F_p that makes the commutator small [10, 12]. This can be achieved by solving a least-squares problem of the form

$$\min \|[Q_u^{-1}FQ_u^{-1}B^{\mathrm{T}}]_j - Q_u^{-1}B^{\mathrm{T}}Q_p^{-1}[F_p]_j\|_{Q_u} \tag{11}$$

where $\|.\|_{Q_u}$ is the $\sqrt{\underline{x}^{\mathrm{T}}Q_u\underline{x}}$ norm and $[F_p]_j$ represents the jth column of matrix F_p. Solving this problem leads to

$$F_p = Q_p(BQ_u^{-1}B^{\mathrm{T}})^{-1}(BQ_u^{-1}FQ_u^{-1}B^{\mathrm{T}})$$

Substituting this expression into (9) gives an approximation of the Schur complement matrix:

$$BF^{-1}B^{\mathrm{T}} \approx (BQ_u^{-1}B^{\mathrm{T}})(BQ_u^{-1}FQ_u^{-1}B^{\mathrm{T}})^{-1}(BQ_u^{-1}B^{\mathrm{T}}) \tag{12}$$

The preconditioner based on this approximation is known as LSC preconditioner. The preconditioner is expensive if the full velocity mass matrix is used in the preconditioner. Therefore, Q_u is replaced with $\hat{Q}_u$, the diagonal of the velocity mass matrix.

In the LSC preconditioner, the first three steps are used to solve the approximate Schur complement (12). Let the residual during GCR iteration step be given by $r = \begin{bmatrix} r_u \\ r_p \end{bmatrix}$, where r_u and r_p refer to the velocity and pressure part, respectively. Then the preconditioning steps with the LSC

DOI: 10.1002/fld

preconditioner are given by:

LSC preconditioner:

1. Solve $(BB^{\mathrm{T}})p = r_p$, where $(BB^{\mathrm{T}}) = B\hat{Q}_u^{-1}B^{\mathrm{T}}$.
2. Update $r_p = B\hat{Q}_u^{-1}F\hat{Q}_u^{-1}B^{\mathrm{T}}p$.
3. Solve $(BB^{\mathrm{T}})p = -r_p$.
4. Update $r_u = r_u - B^{\mathrm{T}}p$.
5. Solve $Fu = r_u$.

In general the convergence of LSC depends both on the mesh size and on the Reynolds number. According to [24] sometimes there is no h-dependency. Furthermore, the sensitivity to the Reynolds number is only mild. In [27] it is shown that for recirculating flows the convergence clearly depends on both issues. In our experiments we have observed that the Reynolds dependency decreases for finer grids. Results for stabilized elements are reported in [28].

Per iteration LSC is more expensive than PCD since it requires two Poisson solves instead of one, whereas PCD requires two extra operators F_p and A_p on the pressure space including some boundary conditions. Nevertheless, its convergence is better and in the literature it is concluded that LSC is faster than PCD.

3.2. SIMPLE(R) preconditioner

SIMPLE-type methods are frequently used to solve the incompressible Navier–Stokes equations. Originally, SIMPLE has been developed for finite volume and finite difference discretizations [22, 23]. The algorithm is based on the following steps. First the pressure is assumed to be known from the prior iteration. Then the velocity is solved from the momentum equations. The newly obtained velocities do not satisfy the continuity equation since the pressure is only a guess. In the next substeps the velocities and pressures are corrected in order to satisfy the discrete continuity equation.

In this paper we apply SIMPLE-type preconditioners for the Navier–Stokes equations discretized by the finite element method.

The algorithm is derived from the block LU decomposition of the coefficient matrix (3)

$$\begin{bmatrix} F & B^{\mathrm{T}} \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & -BF^{-1}B^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} I & F^{-1}B^{\mathrm{T}} \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{13}$$

The approximation $F^{-1} = D^{-1} = \mathrm{diag}(F)^{-1}$ in the $(2,2)$ and $(1,2)$ block of the L and U block matrices, respectively, leads to the SIMPLE algorithm. Solve

$$\begin{bmatrix} F & 0 \\ B & -BD^{-1}B^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{14}$$

and

$$\begin{bmatrix} I & D^{-1}B^{\mathrm{T}} \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u^* \\ \delta p \end{bmatrix} \tag{15}$$

In the SIMPLE algorithm form, the above two steps are performed recursively.

DOI: 10.1002/fld

SIMPLE algorithm:

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

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

Vuik *et al.* [17] used SIMPLE and its variants as a preconditioner to solve the incompressible Navier–Stokes problem. One iteration of the SIMPLE algorithm with assumption $p^* = 0$ is used as a preconditioner. The preconditioner converges nicely if used in combination with the GCR method. However, the convergence rate suffers from an increase in the number of grid elements and Reynolds number.

A variant of SIMPLE, SIMPLER gives Reynolds-independent convergence. Instead of estimating the pressure p^* in the SIMPLE algorithm, p^* is obtained from solving a subsystem

$$\hat{S}p^* = r_p - BD^{-1}((D-F)u^k + r_u) \tag{16}$$

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. In the literature, the SIMPLER algorithm is formulated such that the steps of the algorithm are closely related to the symmetric block Gauss–Seidal method [29]. This form of the SIMPLER preconditioner can be written as

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} u^k \\ p^k \end{pmatrix} + M_L^{-1} B_L \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^k \\ p^k \end{pmatrix} \right) \tag{17}$$

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix} \right) \tag{18}$$

where A represents the complete matrix given in (3), u^k and p^k in (17) are obtained from the previous step (both zero in our case) and

$$B_R = \begin{pmatrix} I & -D^{-1}B^{\mathrm{T}} \\ 0 & I \end{pmatrix}, \quad M_R = \begin{pmatrix} F & 0 \\ B & \hat{S} \end{pmatrix} \tag{19}$$

and

$$B_L = \begin{pmatrix} I & 0 \\ -BD^{-1} & I \end{pmatrix}, \quad M_L = \begin{pmatrix} F & B^{\mathrm{T}} \\ 0 & \hat{S} \end{pmatrix} \tag{20}$$

The steps given in (17) and (18) contain two Poisson solves, two velocity subproblems solves—opposed to one velocity solve in the classical algorithm—and matrix vector updates. However, the extra velocity solve in formulation (17) and (18) has no significant effect on the convergence with the SIMPLER preconditioner.

DOI: 10.1002/fld

Lemma
In the SIMPLER preconditioner/algorithm, both variants (one or two velocity solves) are identical.

Proof
We first start with the choice u^k and p^k which are zero vectors. To solve the system $Pz=r$, (17) reduces to

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = M_L^{-1} B_L \begin{pmatrix} r_u \\ r_p \end{pmatrix} \tag{21}$$

Rewriting (21) leads to

$$p^* = \hat{S}^{-1}(rp - BD^{-1}r_u) \tag{22}$$

and

$$u^* = F^{-1}(r_u - B^{\mathrm{T}}p^*) \tag{23}$$

Next we consider (18). First compute the residual part

$$\begin{pmatrix} r_{un} \\ r_{pn} \end{pmatrix} = \begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix} \tag{24}$$

The velocity part becomes

$$r_{un} = r_u - Fu^* - B^{\mathrm{T}}p^*$$

If we substitute u^* from (23) into r_{un} we get

$$r_{un} = r_u - FF^{-1}(r_u - B^{\mathrm{T}}p^*) - B^{\mathrm{T}}p^* = 0$$

and the pressure part

$$r_{pn} = r_p - B^{\mathrm{T}}u^*$$

therefore, (18) reduces to

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \begin{pmatrix} 0 \\ r_{pn} \end{pmatrix} \tag{25}$$

The formulation (21) and (25) gives rise to three steps in the SIMPLER preconditioner because there is no need to perform an extra velocity solve in (25) when the right-hand side is zero.

$$\delta p = \hat{S}^{-1}(rp - Bu^*) \tag{26}$$

$$u^{k+1} = u^* + BD^{-1}\delta p \tag{27}$$

and

$$p^{k+1} = p^* + \delta p \tag{28}$$

This also holds in the SIMPLER algorithm with non-zero u^k and p^k, which proves the theorem. □

DOI: 10.1002/fld

We observe in our numerical experiments that both variants are different if inexact solves are used, but the convergence of both variants is nearly the same. SIMPLER is more expensive than SIMPLE. One iteration of the SIMPLER algorithm is approximately 1.3 times more expensive than the SIMPLE iteration [17]. SIMPLER convergence is also faster than that of the SIMPLE preconditioner. However, convergence of both preconditioners is h-dependent.

3.3. Improvements in the SIMPLER preconditioner

Since we use different preconditioners, this can only be combined with a flexible Krylov method.

Next we suggest two changes in the SIMPLER preconditioner to improve the convergence for Stokes and Navier–Stokes. Two new changes are suggested in the SIMPLER preconditioner that makes the preconditioner more attractive to use in solving the Navier–Stokes problem.

3.3.1. hSIMPLER. We have observed that in the Stokes problem, the SIMPLER preconditioner shows stagnation at the start of the iterative method. This behavior is not seen in the SIMPLE preconditioner. A better convergence can be achieved if the first iteration is carried out with the SIMPLE preconditioner and after that SIMPLER is employed. We call this combination as hybrid SIMPLER (hSIMPLER). This implementation gives a fair reduction in the number of iterations if the Stokes problem is solved. However, in the Navier–Stokes problem, SIMPLER performs better than hSIMPLER. More details are given in the part with numerical experiments.

3.3.2. MSIMPLER. Elman *et al.* [12, 24] discussed relations between SIMPLE and commutator preconditioners. The more general form of (12) is given by

$$(BF^{-1}B^{\mathrm{T}})^{-1} \approx F_p(BM_1^{-1}B^{\mathrm{T}})^{-1} \tag{29}$$

where

$$F_p = (BM_2^{-1}B^{\mathrm{T}})^{-1}(BM_2^{-1}FM_1^{-1}B^{\mathrm{T}})$$

where M_1 and M_2 are scaling matrices. Consider a new block factorization preconditioner in which the Schur complement is based on a commutator approximation but built on SIMPLE's approximate block factorization written as:

$$P = \begin{bmatrix} F & 0 \\ B & -BM_1^{-1}B^{\mathrm{T}} \end{bmatrix} \begin{bmatrix} I & D^{-1}B^{\mathrm{T}} \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & F_p^{-1} \end{bmatrix} \tag{30}$$

When $M_1 = D$ and F_p is the identity matrix, then the preconditioner formulation (30) corresponds to SIMPLE. The formulation given in (30) is equivalent to the SIMPLE algorithm if the subsystem for the pressure part in step 2 in the SIMPLE algorithm is solved with the approximation given in (29)

$$\hat{S}\delta p = r_p - Bu^*$$

where

$$\hat{S} = -(BM_1^{-1}B^{\mathrm{T}})F_p^{-1}$$

DOI: 10.1002/fld

When FD^{-1} is close to identity, F_p will also be close to identity. This is true in a time-dependent problem with small time steps where the diagonal of F has larger entries than the off-diagonal entries [12].

Here, we utilize the observation of Elman regarding the time-dependent problem. We know that in time-dependent problems

$$F_t = \frac{1}{\Delta t} Q_u + F \tag{31}$$

where F_t represents the velocity matrix for the time-dependent problem and Δt represents the time step. For small time step $F_t \approx (1/\Delta t) Q_u$. This kind of approximation has been used in fractional step methods for solving the unsteady Navier–Stokes problem [30–32]. We use this idea in solving the steady Navier–Stokes problem. Therefore, we choose $M_1 = M_2 = \hat{Q}_u$ in (29) resulting in:

$$F_p = (B\hat{Q}_u^{-1}B^{\mathrm{T}})^{-1}(B\hat{Q}_u^{-1}F\hat{Q}_u^{-1}B^{\mathrm{T}})$$

If we assume that the factor $F\hat{Q}_u^{-1}$ in F_p is close to identity, then

$$F_p = (B\hat{Q}_u^{-1}B^{\mathrm{T}})^{-1}(B\hat{Q}_u^{-1}B^{\mathrm{T}}) \approx I$$

and the approximation (29) becomes

$$BF^{-1}B^{\mathrm{T}} \approx (B\hat{Q}_u^{-1}B^{\mathrm{T}}) \tag{32}$$

Based on this result, we replace D^{-1} in the SIMPLER algorithm by $\hat{Q}_u^{-1}$. We refer to this method as MSIMPLER. MSIMPLER is described by the following step:

MSIMPLER preconditioner:

1. Solve $\hat{S}p^* = r_p - B\hat{Q}_u^{-1}r_u$.
2. Solve $Fu^* = r_u - B^{\mathrm{T}}p^*$.
3. Solve $\hat{S}\delta p = r_p - Bu^*$.
4. Update $u = u^* - \hat{Q}_u^{-1}B^{\mathrm{T}}\delta p$.
5. Update $p = p^* + \delta p$.

In case of quadrilaterals and hexahedrons, $\hat{Q}_u^{-1}$ is the lumped velocity mass matrix, which can also be constructed by using a Newton–Cotes integration rule. In case of quadratic triangles this matrix is singular and we replace it by the diagonal of the consistent mass matrix. For quadratic tetrahedra the situation is somewhat surprising. The lumped velocity mass matrix contains negative entries for the elements corresponding to vertices. Nevertheless, the diagonal elements of $B\hat{Q}_u^{-1}B^{\mathrm{T}}$ are all positive. The convergence of MSIMPLER appears to be comparable to that of hexahedra. If we replace $\hat{Q}_u^{-1}$ by the diagonal of the consistent mass matrix, with positive elements only, the convergence of MSIMPLER becomes much slower.

3.4. Cost comparison of the preconditioners

From a construction point of view, the LSC and MSIMPLER preconditioner are built from available matrices. Another advantage is that the Schur complement is constructed once—at the start of the

DOI: 10.1002/fld

linearization—because $\hat{Q}_u^{-1}$ remains the same during the linearization steps. The preconditioning steps with both preconditioners involve two Poisson solves and one velocity subproblem solve as the major steps.

Computationally, per iteration, the MSIMPLER preconditioner is less expensive than the LSC preconditioner. Both preconditioners have to solve three subsystems (2 for the pressure and 1 for the velocity) per iteration. We assume that solving the subsystem corresponding to the pressure takes sp flops and the subsystem corresponding to the velocity part takes fu flops. nnz_B are the number of non-zero entries in B and nnz_F are the number of non-zero entries in F. Then the cost of the MSIMPLER preconditioner per step is

$$\text{cost}_{\text{msimpler}} = 8nnz_B + 5n + 2m + 2sp + fu$$

and the cost of the LSC preconditioner is

$$\text{cost}_{\text{lsc}} = 6nnz_B + 2nnz_F + 3n + 2sp + fu$$

We assume that the cost of solving the subsystem in both preconditioners is the same. Then the difference in cost of both preconditioners consists of matrix vector multiplications and updates. Per iteration, the difference in cost is:

$$\text{diff} = (2nnz_F) - (2m + 2n + 2nnz_B)$$

If diff>0, MSIMPLER is cheaper than LSC and this appears to be true in finite elements that satisfy the LBB conditions.

4. NUMERICAL EXPERIMENTS

Numerical experiments are performed for the following benchmark problems in 2D and 3D:

1. The 2D L-shaped domain known as the backward facing step is shown in Figure 1. A Poisseuille flow profile is imposed on the inflow ($x=-1;\ 0\leqslant y\leqslant 1$) 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.
2. 2D-driven cavity problem; flow in a square cavity with enclosed boundary conditions and a lid moving from left to right given as

$$u_x = 1 - x^4 \quad \text{at } y = 1,\ -1 \leqslant x \leqslant 1$$

known as regularized cavity problem.

The above two problems are also solved in 3D. Preconditioned Krylov subspace methods are used to solve the Stokes and the Navier–Stokes problem. We divide the experiments into two sections; Section 4.1 which deals only with SIMPLE-type preconditioners and Section 4.2 which consists of a comparison of SIMPLE-type preconditioners with the LSC and SILU preconditioner for various problems. To solve the subsystems iteratively, MG and ILU preconditioned Krylov subspace methods are used. The iteration is stopped if the linear systems satisfy $\|r^k\|_2/\|b\|_2 \leqslant \text{tol}$, where r^k is the residual at the kth step of Krylov subspace method, b is the right-hand side and tol is the desired tolerance value. Some abbreviations used are: t_s for time in seconds, *iter.* for outer iterations, NC for no convergence, in-it-u and in-it-p for the number of iterations taken by the

DOI: 10.1002/fld

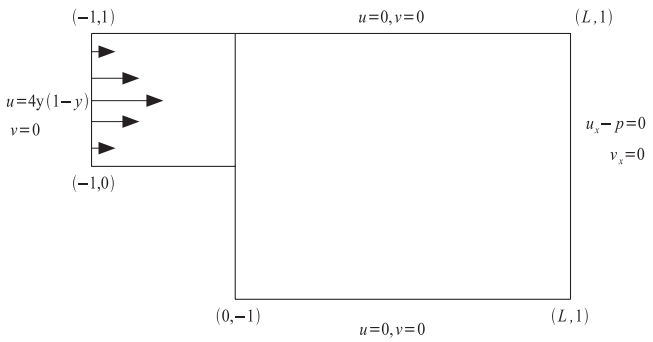

Figure 1. Backward facing step domain.

Table I. Stokes backward facing step solved with preconditioned GCR(20) with accuracy of 10^{-6}, PCG used as an inner solver (SEPRAN).

	SIMPLE		SIMPLER		hSIMPLER		MSIMPLER	
Grid	iter. (ts)	$\frac{\text{in-it-}u}{\text{in-it-}p}$	iter. (ts)	$\frac{\text{in-it-}u}{\text{in-it-}p}$	iter. (ts)	$\frac{\text{in-it-}u}{\text{in-it-}p}$	iter. (ts)	$\frac{\text{in-it-}u}{\text{in-it-}p}$
8×24	39 (0.06) 37 (0.14)	$\frac{64}{299}$	26 (0.05) 19 (0.07)	$\frac{60}{416}$	19 (0.03) 17 (0.06)	$\frac{43}{300}$	11 (0.02) 12 (0.05)	$\frac{18}{164}$
16×46	72 (0.6) 68 (1.94)	$\frac{205}{1032}$	42 (0.5) 30 (0.86)	$\frac{177}{1233}$	31 (0.34) 24 (0.68)	$\frac{124}{907}$	12 (0.1) 15 (0.44)	$\frac{24}{346}$
32×96	144 (8.2) 117 (34)	$\frac{692}{4084}$	NC 114 (32)		44 (5.97) 37 (10.6)	$\frac{692}{2824}$	16 (0.9) 20 (5.75)	$\frac{54}{864}$
64×192	256 (93) 230 (547)	$\frac{2054}{13075}$	NC NC		89 (141) 68 (161)	$\frac{4362}{12033}$	23 (8.5) 25 (60)	$\frac{145}{2307}$

solver to solve subsystems in the preconditioners corresponding to the velocity and pressure part, respectively. SEPRAN[‡] (written in FORTRAN) and IFISS package[§] (written in MATLAB) are used to solve the problems. Numerical experiments are performed on a Intel 2.66 GHz processor with 8 GB RAM.

4.1. SIMPLE-type preconditioners

To test the SIMPLE-type preconditioners for Stokes, we used the first (Stokes) step in the 2D backward facing step problem. All SIMPLE preconditioners are combined with the GCR method [33], since this method allows variable preconditioners. Table I shows the computation time and number of iterations for a series of grids. For each grid, experiments are performed twice, first with low inner accuracy (upper row) and then with high inner accuracy (lower row).

[‡]http://ta.twi.tudelft.nl/sepran/sepran.html.
[§]http://www.maths.manchester.ac.uk.

DOI: 10.1002/fld

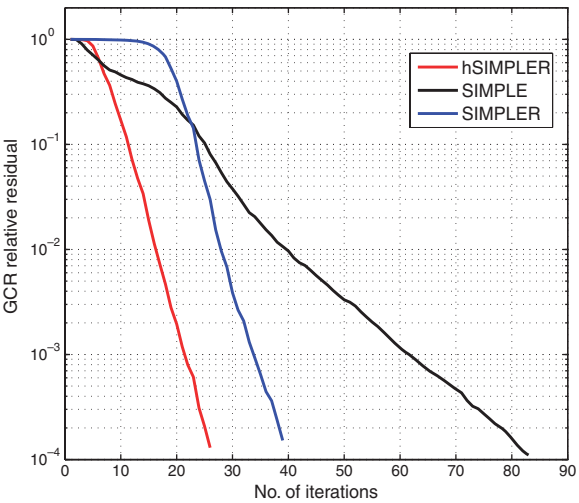

Figure 2. Convergence plot of SIMPLE-type peconditioners for the Stokes problem.

In case of SIMPLE and MSIMPLER, a low inner accuracy implies that the subsystems for the velocities are solved with a relative accuracy of 10^{-1}, and the systems for the pressure with an accuracy of 10^{-2}. In case of high accuracy we used 10^{-6} both for the velocity and pressure. For SIMPLER and hSIMPLER it was necessary to increase the accuracy for the inner solves for increasing grid size. Otherwise, no convergence could be reached. The reason to compare low and high inner accuracies is to investigate the dependence of the SIMPLE-type preconditioners on the inner accuracy. From Table I it is clear that MSIMPLER is the best choice both with respect to the number of iterations as to the CPU time. Increasing the inner accuracy has only a small effect on the number of GCR iterations but a considerable negative effect on the CPU time.

Figure 2 shows that SIMPLER stagnates at the start of iterations. This behavior has been erased by using hSIMPLER. The necessary increase of inner accuracies for finer grids for SIMPLER and hSIMPLER is visible in the smaller difference between upper and lower row in Table I. We also see that SIMPLER does not converge at all for the fine grids.

The behavior of these preconditioners for a Navier–Stokes flow (driven cavity) is shown in Figure 3. A fixed 64×64 grid is used and $Q2-Q1$ elements. The Reynolds number is varied from 100 to 1000 and no upwinding is applied. In all cases the low inner accuracy of the upper row in Table I is used. The left-hand figure shows the average number of inner iterations per Picard step and the right-hand figure shows the overall CPU time, which increases due to the increase of Picard iterations when Reynolds increases. We see that the average number of inner iterations per Picard step depends mildly on the Reynolds number. Again MSIMPLER proves to be superior to the other SIMPLE-type preconditioners.

Based on this conclusion we shall only use MSIMPLER in the next sections to compare it with other types of preconditioners.

4.2. Preconditioners comparison

In this section, we compare MSIMPLER with LSC and SILU. First we consider the 2D lid-driven cavity and the 2D backward facing step with $Q2$–$Q1$ rectangular elements. Next we investigate

DOI: 10.1002/fld

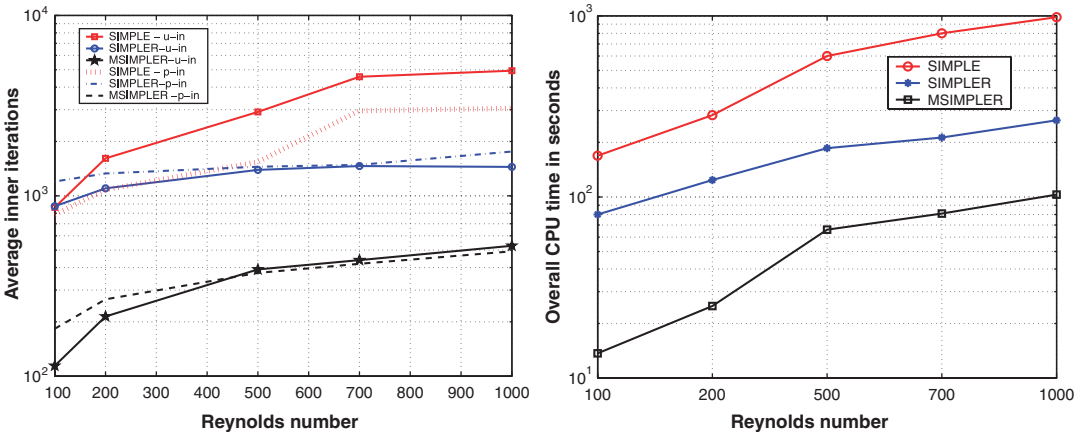

Figure 3. The Navier–Stokes problem solved in $Q2$–$Q1$ discretized 64×64 driven cavity problem with varying Reynolds number, number of average inner iterations (left), CPU time in seconds (right)-(SEPRAN).

Table II. Backward facing step Navier–Stokes problem with preconditioned Bi-CGSTAB with accuracy 10^{-6} (MG solver is used to solve subsystems (IFISS)).

	$Re = 100$		$Re = 200$		$Re = 400$	
	LSC	MSIMPLER	LSC	MSIMPLER	LSC	MSIMPLER
Grid	iter. (t_s)					
16×48	17 (8)	9 (4.5)	27 (13)	15 (7)	73 (39)	29 (16)
32×96	16 (17)	11 (13.7)	15 (22)	10 (17)	24 (28.5)	15 (21)
64×192	24 (119)	20 (99)	23 (118)	15 (84)	22 (112)	18 (102)

the behavior for 3D problems using both hexahedra and tetrahedra. Finally, we test the methods for 2D stretched grids.

4.2.1. Comparison in 2D. The first test we apply is the solution of the 2D backward facing step for various grid sizes and Reynolds numbers. Table II shows the number of iterations and CPU time in the second Picard step both for LSC and MSIMPLER. The system of equations for pressure and velocity is solved by one MG cycle. Although we see that the convergence depends on the Reynolds number for coarse grids, this is no longer the case for the finest grid. Furthermore, it is clear that the number of iterations decreases for fixed Reynolds number for finer grids. Presumably this is due to the decrease in cell Reynolds number (*element size*$/\nu$). In all cases MSIMPLER requires less iterations than LSC, but the difference becomes small for increasing mesh size. The relative good result of the last number in the MSIMPLER column must be because of the better cell Reynolds number.

Table III shows the convergence of MSIMPLER, LSC and SILU in case we replace MG by an ILU preconditioned BI-CGSTAB solver. The accuracy for the inner solves is 10^{-2}, which is sufficient to reach the final accuracy of the Navier–Stokes problem without increasing the number

DOI: 10.1002/fld

Table III. Backward facing step: preconditioned GCR is used to solve the Navier–Stokes problem with accuracy 10^{-2}, using Bi-CGSTAB as inner solver, the number of iterations is the accumulated iterations consumed by the outer and inner solvers (SEPRAN).

	LSC	MSIMPLER	SILU(Bi-CGSTAB)
Grid	iter. (t_s) $\frac{\text{in-it-}u}{\text{in-it-}p}$		iter. (t_s)
$Re=100$ (11 Picard iterations)			
16×48	114 (1.7) $\frac{464}{2609}$	73 (1) $\frac{213}{1308}$	246 (0.8)
32×96	193 (22) $\frac{1928}{8598}$	106 (10.5) $\frac{859}{3238}$	731 (8.7)
64×192	328 (545) $\frac{9703}{26884}$	182 (162) $\frac{4088}{11266}$	2071 (95)
128×384	695 (8863) $\frac{55000}{154000}$	296 (2806) $\frac{16000}{54000}$	6352 (1155)
$Re=200$ (17 Picard iterations)			
16×48	179 (2.3) $\frac{549}{3320}$	137 (1.7) $\frac{332}{2546}$	436 (1.3)
32×96	302 (31) $\frac{2518}{12611}$	161 (14) $\frac{1052}{4619}$	1100 (13)
64×192	598 (983) $\frac{13947}{44840}$	232 (191) $\frac{4718}{12761}$	3114 (141)
128×384	946 (10405) $\frac{62000}{203000}$	541 (6301) $\frac{35000}{63000}$	2668 (9038)
$Re=400$ (31 Picard iterations)			
16×48	441 (4.93) $\frac{1004}{7600}$	356 (3.9) $\frac{734}{6445}$	716 (2.13)
32×96	528 (51) $\frac{3000}{29000}$	328 (25) $\frac{1593}{9764}$	1706 (20.7)
64×192	NC	405 (408) $\frac{5130}{19299}$	5366 (246)
128×384	NC	663 (7025) $\frac{29600}{66480}$	NC

of Picard iterations. In this table we report the sum of the iterations in all Picard steps, which gives a complete picture of the whole problem. In this case the difference between MSIMPLER and LSC is much more pronounced. The reason must be the change of inner solver. Furthermore, we see that SILU is faster than MSIMPLER except for the finest grid in combination with the larger Reynolds numbers.

In order to see if the block preconditioners are sensitive to the inner accuracies, we compare one MG cycle for the inner solver with an exact inner solver in Table IV. From this table it is clear that MSIMPLER is hardly affected by the inner accuracy, whereas LSC is more sensitive in case of coarse grids in combination with a high Reynolds number.

4.2.2. Comparisons in 3D. Iterative solvers for the Navier–Stokes are especially important for 3D problems. In our experiments, we used both hexahedra and tetrahedra. Only Taylor–Hood elements have been applied.

Table V gives the results of the various preconditioners for the Stokes problem solved on a 3D backward facing step with hexahedral elements. An IC preconditioned CG solver is used as inner solver for the block preconditioners. MSIMPLER requires the least number of iterations and shows almost grid independent convergence behavior. The computation time of SILU is also good, but for finer grids it becomes more expensive than MSIMPLER.

DOI: 10.1002/fld

Table IV. Driven cavity flow problem: The Navier–Stokes problem is solved with preconditioned Bi-CGSTAB with accuracy 10^{-6} (MG and direct solver are used to solve subsystems (IFISS)).

	$Re=100$		$Re=500$		$Re=1000$	
	LSC	MSIMPLER	LSC	MSIMPLER	LSC	MSIMPLER
Grid	MG/Exact	MG/Exact	MG/Exact	MG/Exact	MG/Exact	MG/Exact
No. of iterations per Picard step						
16×16	$\frac{14}{10}$	$\frac{10}{9}$	$\frac{53}{29}$	$\frac{28}{25}$	$\frac{102}{55}$	$\frac{50}{53}$
32×32	$\frac{19}{15}$	$\frac{13}{12}$	$\frac{35}{26}$	$\frac{26}{20}$	$\frac{82}{55}$	$\frac{45}{43}$
64×64	$\frac{22}{22}$	$\frac{19}{19}$	$\frac{34}{29}$	$\frac{25}{25}$	$\frac{63}{55}$	$\frac{34}{32}$
128×128	$\frac{27}{27}$	$\frac{25}{28}$	$\frac{47}{44}$	$\frac{42}{44}$	$\frac{62}{59}$	$\frac{43}{43}$

Table V. 3D backward facing step (hexahedra): The Stokes problem is solved with accuracy 10^{-6} (PCG is used as inner solver in the block preconditioners (SEPRAN)).

	SIMPLE	LSC	MSIMPLER	SILU (Bi-CGSTAB)
Grid		iter. $(t_s)\,\frac{\text{in-it-}u}{\text{in-it-}p}$		iter. (t_s)
$8\times8\times16$	44 (4) $\frac{97}{342}$	16 (1.9) $\frac{41}{216}$	14 (1.4) $\frac{28}{168}$	26 (0.7)
$16\times16\times32$	84 (107) $\frac{315}{1982}$	29 (51) $\frac{161}{1263}$	17 (21) $\frac{52}{766}$	65 (16.7)
$24\times24\times48$	99 (447) $\frac{339}{3392}$	26 (233) $\frac{193}{2297}$	17 (77) $\frac{46}{1116}$	117 (118)
$32\times32\times40$	132 (972) $\frac{574}{5559}$	37 (379) $\frac{233}{2887}$	20 (143) $\frac{66}{1604}$	189 (235)

In the Navier–Stokes problem it is sufficient to use an accuracy of 10^{-2} per Picard step. In this case SILU performs slightly better than MSIMPLER (see Table VI).

To investigate the behavior of the preconditioners for tetrahedral elements, we solved the 3D lid-driven cavity problem (Table VII). For the Stokes problem the result is comparable to the hexahedral case. MSIMPLER requires less CPU time than LSC and SILU. The number of GCR iterations is almost mesh independent.

The situation for Navier–Stokes is different from that of hexahedra. Table VIII gives the CPU time and the number of iterations. Now MSIMPLER proves to be the best choice. The increase in iterations for increasing Reynolds number is caused by an increase in the number of Picard iterations. The Reynolds dependency of all methods per Picard iteration is only mild.

4.2.3. Grid stretching. One of the unsolved issues in the iterative solution of the Navier–Stokes equations is the case of stretched grids. In practical applications it is very common to have stretched grids; hence, it is important that an iterative solver is capable of dealing with such meshes. Therefore, 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, see Figure 4. The stretch factor SF is defined as the ratio of the largest and smallest edge in the grid.

DOI: 10.1002/fld

Table VI. 3D backward facing step (hexahedra): The Navier–Stokes problem is solved with the accuracy 10^{-4}, a linear system at each Picard step is solved with accuracy 10^{-2} using preconditioned Krylov subspace methods (Bi-CGSTAB is used as inner solver in block preconditioners (SEPRAN)).

	LSC	MSIMPLER	SILU
Re	GCR iter. (t_s)	GCR iter. (t_s)	Bi-CGSTAB iter. (t_s)
$8\times8\times16$			
100	117 (17.6)	74 (9.6)	140 (8.9)
200	176 (25)	112 (14.8)	255 (13.8)
400	280 (36)	168 (21)	1688 (49)
$16\times16\times32$			
100	173 (462)	96 (162)	321 (114)
200	256 (565)	145 (223)	461 (173)
400	399 (745)	235 (312)	768 (267)
$32\times32\times40$			
100	240 (5490)	130 (1637)	1039 (1516)
200	NC	193 (2251)	1378 (2000)
400	675 (11000)	295 (2800)	1680 (2450)

Table VII. 3D Lid-driven cavity problem (tetrahedra): The Stokes problem is solved with accuracy 10^{-6} (PCG is used as inner solver in block preconditioners (SEPRAN)).

	LSC	MSIMPLER	SILU (Bi-CGSTAB)
Grid	iter. (t_s) $\frac{\text{in-it-}u}{\text{in-it-}p}$		iter. (t_s)
$8\times8\times8$	9 (0.24) $\frac{17}{52}$	8 (0.23) $\frac{16}{53}$	32 (0.25)
$16\times16\times16$	12 (4.8) $\frac{49}{152}$	11 (3.4) $\frac{31}{150}$	73 (5.6)
$32\times32\times32$	17 (89) $\frac{129}{426}$	14 (54) $\frac{68}{380}$	237 (162)

Table VIII. 3D lid-driven cavity problem (tetrahedra): The Navier–Stokes problem is solved with accuracy 10^{-4}, a linear system at each Picard step is solved with accuracy 10^{-2} using preconditioned Krylov subspace methods (Bi-CGSTAB is used as inner solver in block preconditioners (SEPRAN)).

	LSC	MSIMPLER	SILU
Re	GCR iter. (t_s)	GCR iter. (t_s)	Bi-CGSTAB iter. (t_s)
$16\times16\times16$			
20	30 (20)	20 (16)	144 (22)
50	57 (37)	37 (24)	234 (35)
100	120 (81)	68 (44)	427 (62)
$32\times32\times32$			
20	38 (234)	29 (144)	463 (353)
50	87 (544)	53 (300)	764 (585)
100	210 (1440)	104 (654)	1449 (1116)

DOI: 10.1002/fld

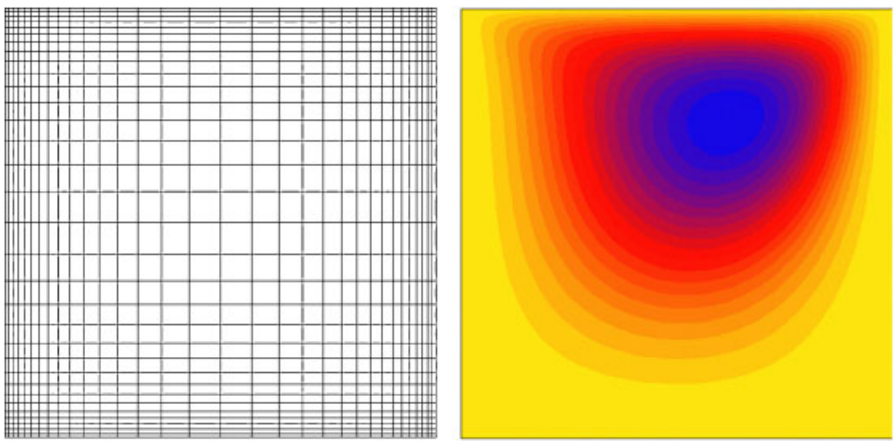

Figure 4. A 32×32 grid with stretch factor $= 8$ (left), Streamlines plot on the stretched grid (right)-(SEPRAN).

Table IX. 2D Lid-driven cavity problem on 64×64 stretched grid: The Stokes problem is solved with accuracy 10^{-6} (PCG is used as inner solver in block preconditioners (SEPRAN)).

	LSC	MSIMPLER	SILU
Stretch factor	GCR iter.	GCR iter.	Bi-CGSTAB iter.
1	20	17	96
8	49	28	189
16	71	34	317
32	97	45	414
64	145	56	NC
128	NC	81	NC

Results for solving the Stokes problem are shown in Table IX. The convergence of MSIMPLER, LSC and SILU detoriates with an increase in the stretching factor. All three preconditioners show an increase in the number of iterations with increase in stretching. For a large stretch factor, the preconditioners even fail to converge.

Compared with the performance of these preconditioners in the Stokes problem, the situation is even worse in the Navier–Stokes problem. Until now, as far as we know no results for LSC in stretched grids are published. In Table X, we see that MSIMPLER and LSC perform poorly in stretched grids for the Navier–Stokes problem. With the increase in stretching factor, all preconditioners mentioned show bad convergence. The inner and outer iterations in the block preconditioners stagnate after some reduction in the residual. For a certain stretch factor, the performance of these preconditioners becomes worse with the increase in the Reynolds number and the grid size.

5. CONCLUSIONS

In this paper we have studied the convergence behavior of some block preconditioners for Stokes and Navier–Stokes problems both in 2D and 3D. Results for various grid sizes and Reynolds

DOI: 10.1002/fld

Table X. 2D lid-driven cavity problem on stretched grid: The Navier–Stokes problem is solved with accuracy 10^{-4}. A linear system in each Picard step is solved with accuracy 10^{-2} using preconditioned Krylov subspace methods (Bi-CGSTAB is used as inner solver in the block preconditioners (SEPRAN)).

		LSC	MSIMPLER	SILU
Stretch factor	Re	GCR iter.	GCR iter.	Bi-CGSTAB iter.
32×32				
4	100	148	86	181
	200	214	149	247
	400	NC	261	268
8	100	171	104	242
	200	228	155	234
	400	NC	307	311
64×64				
4	100	NC	114	526
	200	NC	179	591
	400	NC	407	630
8	100	NC	141	1131
	200	NC	233	754
	400	NC	NC	814

numbers have been investigated. In some cases we also compared the convergence with an algebraic preconditioner (SILU). We come to the following conditions:

- The performance of SIMPLER in solving the Stokes problem can be enhanced by employing the first iteration with SIMPLE and then use SIMPLER (hSIMPLER).
- MSIMPLER is at present the fastest of all SIMPLE-type preconditioners.
- In contrast with SIMPLER, MSIMPLER is not sensitive to the accuracies that are used for the inner solvers.
- MSIMPLER is the cheapest to construct of all SIMPLE-type methods since the Schur complement matrix is constant during the nonlinear steps and can therefore be made at the start of the process. This is because the scaling is independent of the velocity.
- In all our experiments MSIMPLER proved to be cheaper than LSC. This concerns both the number of outer iterations, inner iterations and CPU time.
- The number of outer iterations in MSIMPLER hardly increases if a direct solver for the subsystems is replaced by an iterative solver. This is in contrast with LSC where large differences are observed. It appears that the combination of LSC with MG is almost optimal, but the combination of LSC with a PCG inner solver can take many iterations and much CPU time.
- In our experiments, MSIMPLER proved to be cheaper than SILU, especially when the problem is solved with high accuracy.
- The performance of all these preconditioners is affected by grid stretching. The number of iterations increases with an increase in stretching or even diverges in some cases.

We observed that MSIMPLER is able to deal with grids having small aspect ratio, both in 2D and 3D. However, in case of high aspect ratio, MSIMPLER fails to converge. Unfortunately, all other preconditioners discussed in this paper have the same behavior. Thus, further research to solve this problem is needed.

DOI: 10.1002/fld

REFERENCES

1. Wille SØ, Loula AFD. A priori pivoting in solving the Navier–Stokes equations. *Communications in Numerical Methods in Engineering* 2002; **18**(10):691–698.
2. 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.
3. 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.
4. Duff IS, Meurant GA. The effect of ordering on preconditioned conjugate gradients. *BIT* 1989; **29**(4):635–657.
5. 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.
6. 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.
7. Benzi M, Tuma M. A sparse approximate inverse preconditioner for nonsymmetric linear systems. *SIAM Journal on Scientific Computing* 1998; **19**(3):968–994.
8. Bollhofer M, Saad Y. Multilevel preconditioners constructed from inverse-based ILUs. *SIAM Journal on Scientific Computing* 2006; **27**(5):1627–1650.
9. ur Rehman M, Vuik C, Segal G. A comparison of preconditioners for incompressible Navier–Stokes solvers. *International Journal for Numerical Methods in Fluids* 2008; **57**:1731–1751. DOI: 10.1002/fld.1684.
10. Elman HC. Preconditioning for the steady-state Navier–Stokes equations with low viscosity. *SIAM Journal on Scientific Computing* 1999; **20**(4):1299–1316.
11. Benzi M, Olshanskii MA. An augmented Lagrangian-based approach to the Oseen problem. *SIAM Journal on Scientific Computing* 2006; **28**(6):2095–2113.
12. 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.
13. Gauthier A, Saleri F, Veneziani A. A fast preconditioner for the incompressible Navier–Stokes equations. *Computing and Visualization in Science* 2004; **6**(2):105–112. DOI: 10.1007/s00791-003-0114-z.
14. 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.
15. de Niet AC, Wubs FW. Two preconditioners for saddle point problems in fluid flows. *International Journal for Numerical Methods in Fluids* 2007; **54**(4):355–377.
16. Olshanskii MA. An iterative solver for the Oseen problem and numerical solution of incompressible Navier–Stokes equations. *Numerical Linear Algebra with Applications* 1999; **6**:353–378.
17. 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.
18. Manguoglu M, Sameh AH, Tezduyar TE, Sathe S. A nested iterative scheme for computation of incompressible flows in long domains. *Journal of Computational Mechanics* 2008; **43**:73–80. DOI: 10.1007/s00466-008-0276-0.
19. Manguoglu M, Sameh AH, Saied F, Tezduyar TE, Sathe S. Preconditioning techniques for nonsymmetric linear systems in the computation of incompressible flows. *Journal of Applied Mechanics*, in press.
20. Benzi M, Golub GH, Liesen J. Numerical solution of saddle point problems. *Acta Numerica* 2005; **14**:1–137.
21. Benzi M. Preconditioning techniques for large linear systems: a survey. *Journal of Computational Physics* 2002; **182**(2):418–477.
22. Wesseling P. *Principles of Computational Fluid Dynamics*. Springer Series in Computational Mathematics, vol. 29. Springer: Berlin, Heidelberg, 2001.
23. Patankar SV. *Numerical Heat Transfer and Fluid Flow*. McGraw-Hill: New York, 1980.
24. Elman HC, Silvester D, Wathen AJ. *Finite Elements and Fast Iterative Solvers with Applications in Incompressible Fluids Dynamics*. Oxford University Press: Oxford, 2005.
25. Sloan SW. An algorithm for profile and wavefront reduction of sparse matrices. *International Journal for Numerical Methods in Engineering* 1986; **23**(2):239–251.
26. Cuthill E, McKee J. Reducing the bandwidth of sparse symmetric matrices. *Proceedings of the 1969 24th National Conference*. ACM Press: New York, 1969; 157–172.
27. Olshanskii MA, Vassilevski YV. Pressure Schur complement preconditioners for the discrete Oseen problem. *SIAM Journal on Scientific Computing* 2007; **29**(6):2686–2704.
28. Elman H, Howle VE, Shadid J, Silvester D, Tuminaro R. Least squares preconditioners for stabilized discretizations of the Navier–Stokes equations. *SIAM Journal on Scientific Computing* 2007; **30**(1):290–311.

29. Vuik C, Saghir A. The Krylov accelerated SIMPLE(R) method for incompressible flow. *Report 02-01*, Department of Applied Mathematical Analysis, Delft University of Technology, Delft, 2002.
30. Blair Perot J. An analysis of the fractional step method. *Journal of Computational Physics* 1993; **108**(1):51–58.
31. Blasco J, Codina R, Huerta A. A fractional-step method for the incompressible Navier–Stokes equations related to a predictor-multicorrector algorithm. *International Journal for Numerical Methods in Fluids* 1998; **28**(10): 1391–1419.
32. Chorin AJ. Numerical solution of the Navier–Stokes equations. *Mathematics of Computation* 1968; **22**(104): 745–762.
33. Eisenstat C, Elman HC, Schultz MH. Variational iterative methods for nonsymmetric systems of linear equations. *SIAM Journal on Numerical Analysis* 1983; **20**(2):345–357.

DOI: 10.1002/fld