DELFT UNIVERSITY OF TECHNOLOGY

REPORT 08-09

SIMPLE-type preconditioners for the Oseen problem

M. ur Rehman, C. Vuik G. Segal

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2008

SIMPLE-TYPE PRECONDITIONERS FOR THE OSEEN PROBLEM

M. UR REHMAN, C. VUIK, AND G. SEGAL

Abstract. In this report, we discuss block preconditioners used to solve the incompressible Navier-Stokes equations. We emphasize on the approximation of the Schur complement used in SIMPLE-type preconditioners. In the usual formulation, the Schur complement uses scaling with the diagonal of the convection diffusion matrix. A variant of SIMPLE, SIMPLER is studied. Convergence of the SIMPLER preconditioner depends on the grid size, but not on the Reynolds number. We introduce a new variant of SIMPLER (Modified SIMPLER or MSIMPLER), based on the diagonal of the velocity mass matrix as scaling matrix instead of the diagonal of the convection-diffusion matrix.

With the new approximation, we observe a drastic improvement in convergence for fairly large problems. MSIMPLER shows better convergence than the well-known least-squares commutator (LSC) preconditioner which is also based on the diagonal of the velocity mass matrix.

Keywords. Navier-Stokes, Block preconditioners, ILU preconditioners, Krylov subspace methods

AMS subject classifications. 65F10, 65F50, 65N22

Contents

List of Figures

List of Tables

1. Introduction

Solution of the incompressible Navier-Stokes problem, numerically, is a hot topic in scientific research nowadays. Solving the resulting linear system efficiently is of prime interest, because most of the CPU time and memory is consumed in solving the systems. The incompressible Navier-Stokes problem is given as

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

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

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

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

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

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

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 $m \leq n$. 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.

In order to solve linear systems of shape (3) we apply Krylov subspace methods. These methods can only be applied in combination with a suitable preconditioner. Two types of preconditioners are distinguished:
Algebraic preconditioners applied on the complete system (3) and physics-based approaches using some kind of block preconditioner. In general algebraic preconditioners are based on ILU factorization of the coefficient matrix. In order to avoid problems with zeros on the main diagonal, either dynamic pivoting or a clever a-priori reordering technique has to be applied [1–9].
In [9] we published an a-priori reordering technique (SILU), that converges fast for small to mid-sized grids.
Characteristics of algebraic preconditioners are:
- they do not require any extra knowledge of the system
- implementation is cheap and straight-forward
- convergence may be slow for fine grids.

Block preconditioners, on the other hand, distinguish subsystems for pressure and velocity separately. A special algorithm takes care of the overall convergence. The preconditioners are based on the knowledge of the pressure and velocity matrices. Examples of such preconditioners can be found in [10–16]. The goal of all these preconditioners is to get convergence independent of the Reynolds number and grid size. In [17] and [18] one can find an overview of recent published block preconditioners.

In Section 2 we shall discuss two types of block preconditioners: block preconditioners based on approximate commutators in particular the least squares commutator (LSC) and various kinds of SIMPLE-type preconditioners.
We propose a variant of SIMPLER (MSIMPLER) that improves the convergence and makes its performance better than LSC. In Section 3, some numerical experiments are presented for two benchmark problems: backward facing step and lid driven cavity flow. In Section 4, we conclude our numerical experiments.

2. Preconditioning strategies based on block factorization

Preconditioning is a technique used to enhance the convergence of an iterative method to solve a large linear system iteratively. Instead of solving a system $Ax = b$, one solves a system $P^{-1}Ax = P^{-1}b$, 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.
Block preconditioners for the (Navier-)Stokes equations are characterized by a segregation of velocity and pressure during each step in the iterative solution procedure. System (3) is solved by a Krylov subspace solver, for example GCR. To accelerate the convergence of this process a preconditioner is applied. This implies that during each iteration an extra system of the shape $Pz = r$ has to be solved. $r = \begin{bmatrix} r_u \\ r_p \end{bmatrix}$, is a residual, where r_u and r_p correspond to the velocity and the pressure part respectively. The preconditioner P is based on the block LDU factorization of (3):

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = LDU = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{4}$$

where $S = -BF^{-1}B^T$, is known as the Schur complement matrix.
Most of the preconditioners published, try to approximate the DU part:

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

Preconditioners based on formulation (5) are known as block triangular preconditioners (P_t). The most expensive part of the block preconditioner is the inverse of F and S. In general, block preconditioners consist of some good and cheap approximations to F^{-1} and S^{-1} along with matrix vectors multiplications and updates.
In order to investigate the spectral properties of the preconditioned matrix one can consider the following generalized eigenvalue problem

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}, \tag{6}$$

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

$$BF^{-1}B^T p = \mu_i S p,$$

where μ_i are the eigenvalues corresponding to the Schur complement [19]. From the eigenvalues, it is evident that convergence with the preconditioners based on formulation (5) depends strongly on the approximation to the Schur complement matrix. In ([11], [10], [13]) one can find examples of such preconditioners.

2.1. Block preconditioners based on approximate commutators.

Kay et al ([13]) published a class of approximations for the Schur compliment based on the assumption that the commutator of the convection diffusion operator on the velocity space, multiplied by the gradient operator on the velocity space, with, the gradient

operator acting on the convection diffusion operator in the pressure space, is small. The discrete commutator in terms of finite element matrices is given as

$$\varepsilon_h = (Q_u^{-1}F)(Q_u^{-1}B^T) - (Q_u^{-1}B^T)(Q_p^{-1}F_p), \tag{7}$$

where Q_u denotes 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 nodal values. Pre-multiplication of (7) 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

$$BF^{-1}B^T \approx BQ_u^{-1}B^TF_p^{-1}Q_p. \tag{8}$$

Kay et al replace the expensive part $BQ_u^{-1}B^T$ by the pressure Laplacian matrix A_p which is spectrally equivalent to $BQ_u^{-1}B^T$. Hence

$$S = -BF^{-1}B^T \approx -A_pF_p^{-1}Q_p. \tag{9}$$

The approximation substituted in (5) is known as pressure convection-diffusion (PCD) preconditioner.
Elman et al [11] created a matrix F_p that minimizes the commutator (7). This can be achieved by solving a least squares problem. For the jth column of matrix F_p, the least squares problem is of the form:

$$min\|[Q_u^{-1}FQ_u^{-1}B^T]_j - Q_u^{-1}B^TQ_p^{-1}[F_p]_j\|_{Q_u}, \tag{10}$$

where $\|.\|_{Q_u}$ is the $\sqrt{\underline{x}^TQ_u\underline{x}}$ norm. The normal equations associated with this problem are:

$$Q_p^{-1}BQ_u^{-1}B^TQ_p^{-1}[F_p]_j = [Q_p^{-1}BQ_u^{-1}FQ_u^{-1}B^T]_j,$$

which leads to the following definition of F_p:

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

Substituting this expression into (8) gives an approximation of the Schur complement matrix:

$$BF^{-1}B^T \approx (BQ_u^{-1}B^T)(BQ_u^{-1}FQ_u^{-1}B^T)^{-1}(BQ_u^{-1}B^T). \tag{11}$$

The preconditioner based on this approximation is known as the Least Squares Commutator (LSC) preconditioner. The preconditioner is expensive if the full velocity mass matrix is used in the preconditioner. Therefore, Q_u^{-1} is replaced with $\hat{Q_u}^{-1}$, the inverse of the diagonal of the velocity mass matrix. According to the literature ([13], [11]), the convergence of the LSC and PCD preconditioners is independent of the grid size, and only mildly dependent on the Reynolds number. The preconditioners have been tested for the driven cavity and backward facing step problem using Q2-Q1 and Q2-P1 elements. It has also been tested for stabilized elements. LSC requires two Poisson per iteration, whereas PCD requires only one. On the other hand, PCD requires two extra operators F_p and A_p on the pressure space. These operators needboundary conditions for the pressure, which are taken to be zero at inflow and Neumann at all other boundaries. So PCD requires extra start-up time, but per iteration LSC is more expensive.

2.2. **SIMPLE(R) preconditioner.** SIMPLE (Semi Implicit Method for Pressure Linked Equations) [20], [21] 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 continuity since the pressure field assumed is only a guess. Corrections to velocities and pressure are proposed to satisfy the discrete continuity equation. The algorithm can be derived from the block LU decomposition of the coefficient matrix (3)

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & -BF^{-1}B^T \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \quad (12)$$

F in (12) is approximated by the diagonal of F, D. This leads to the SIMPLE algorithm

$$\begin{bmatrix} F & 0 \\ B & -BD^{-1}B^T \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \quad (13)$$

and

$$\begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u^* \\ \delta p \end{bmatrix}. \quad (14)$$

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

SIMPLE algorithm:

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

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

Vuik et al [16], 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 gives convergence if combined with the GCR method. However, the convergence rate depends on the number of grid elements and the Reynolds number. The convergence of SIMPLER, a variant of SIMPLE, is independent of Reynolds. Instead of estimating pressure p^* in the SIMPLE algorithm, p^* is obtained from solving a subsystem:

$$\hat{S}p^* = g - BD^{-1}((D - F)u^k + f), \quad (15)$$

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 closely relates the Symmetric Block Gauss-Seidal method [22]. This form of the SIMPLER preconditioner can be written as:

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} u^k \\ p^k \end{pmatrix} + M_L^{-1}B_L \left(\begin{pmatrix} f \\ g \end{pmatrix} - A \begin{pmatrix} u^k \\ p^k \end{pmatrix} \right), \quad (16)$$

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \left(\begin{pmatrix} f \\ g \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix} \right), \quad (17)$$

where A represents the complete matrix given in (3), u^k and p^k in (16) are obtained from the previous step (both zero in our case) and

$$B_R = \begin{pmatrix} I & -D^{-1}B^T \\ 0 & I \end{pmatrix}, \; M_R = \begin{pmatrix} F & 0 \\ B & \hat{S} \end{pmatrix} \text{ and} \quad (18)$$

$$B_L = \begin{pmatrix} I & 0 \\ -BD^{-1} & I \end{pmatrix}, \; M_L = \begin{pmatrix} F & B^T \\ 0 & \hat{S} \end{pmatrix}. \quad (19)$$

The steps given in (16) and (17) contain two Poisson solves, two velocity subproblems solves- posed to one velocity solve in the classical algorithm- and matrix vector updates. However, in practical SIMPLER implementation -per preconditioning step - the velocity solve is performed once because no significant effect on the convergence with the SIMPLER preconditioner is observed, see Appendix-A. In the remainder of this paper, we will use SIMPLER with one velocity solve. One iteration of the

SIMPLER algorithm is approximately 1.3 times more expensive than the SIMPLE iteration [16]. SIMPLER convergence is usually faster than SIMPLE. However, convergence with both preconditioners is decreased with an increase in the number of grid elements.

2.3. **SIMPLE-type preconditioners and commutators preconditioners.** Elman et al [19], [11] discussed relations between SIMPLE and commutator preconditioners. The more general form of (11) is given as:

$$(BF^{-1}B^T)^{-1} \approx F_p(BM_1^{-1}B^T)^{-1}, \tag{20}$$

where

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

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:

$$P = \begin{bmatrix} F & 0 \\ B & -BM1^{-1}B^T \end{bmatrix} \begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} I & 0 \\ 0 & F_p^{-1} \end{bmatrix}. \tag{21}$$

When $M_1 = D$ and F_p the identity matrix, the preconditioner formulation (21) corresponds to SIMPLE. The formulation given in (21) is equivalent to the SIMPLE algorithm if the subsystem for the pressure part in step 3 in the SIMPLE algorithm is solved with the approximation given in (20)

$$\hat{S}\delta p = g - Bu^* \quad \text{where } \hat{S} = -(BM_1^{-1}B^T)F_p^{-1}.$$

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 [11].

2.4. **Improvements in the SIMPLER preconditioner.**

2.4.1. *hSIMPLER.* We have observed that in the Stokes problem, the SIMPLER preconditioner shows stagnation at the start of the iterative method . The 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 then SIMPLER is employed. We call it hSIMPLER(hybrid SIMPLER). 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 section on numerical experiments.

2.4.2. *MSIMPLER.* It has been shown in [11], that scaling with the velocity mass matrix improves the convergence of the LSC preconditioner. Here we utilize the observation of Elman regarding the time dependent problem. We know that in time dependent problems,

$$F_t = \frac{1}{\Delta t}Q_u + F, \tag{22}$$

where F_t represents the velocity matrix for the time dependent problem and Δt represents the time step. For small time steps $F_t \approx \frac{1}{\Delta t}Q_u$. This kind of approximation has been used in fractional step methods for solving the unsteady Navier-Stokes problem [23], [24], [25]. We use this idea in solving the steady Navier-Stokes problem. Therefore, in (20), we assume that $M_1 = M_2 = \hat{Q_u}$ then

$$F_p = (B\hat{Q_u}^{-1}B^T)^{-1}(B\hat{Q_u}^{-1}F\hat{Q_u}^{-1}B^T),$$

we assume that the factor $F\hat{Q_u}^{-1}$ in F_p is close to identity:

$$F_p = (B\hat{Q_u}^{-1}B^T)^{-1}(B\hat{Q_u}^{-1}B^T) = I.$$

Then the approximation (20) becomes

$$BF^{-1}B^T \approx (B\hat{Q_u}^{-1}B^T). \tag{23}$$

A new variant of SIMPLER arises if we replace the scaling with the diagonal of the velocity matrix by the diagonal of the velocity mass matrix. We call it MSIMPLER (modified SIMPLER). The MSIMPLER preconditioner is given by:

MSIMPLER preconditioner:

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

In the LSC preconditioner, the first three steps are used to solve the approximate Schur complement (5). The preconditioning steps with LSC are:

LSC preconditioner:

(1) Solve $BBtp = r_p$ where $BBt = B\hat{Q_u}^{-1}B^T$
(2) Update $r_p = B\hat{Q_u}^{-1}F\hat{Q_u}^{-1}B^T p$.
(3) Solve $BBtp = -r_p$.
(4) update $r_u = r_u - B^T p$.
(5) Solve $Fu = r_u$.

2.5. **Cost comparison of the preconditioners.** From a construction point of view, the LSC and MSIMPLER preconditioners are built from available matrices. Another advantage is, that the Schur complement is constructed once -at the start of the linearization - because $\hat{Q_u}^{-1}$ remains the same in the linearization steps. The preconditioning steps with both preconditioners involves 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 (two for the pressure and one 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 and cost of the matrix vector product is computed by assuming an average number of non-zero entries per row of a matrix. Then the cost of the MSIMPLER preconditioner per step is:

$$cost_{msimpler} = 8nnz_B + 5n + 2m + 2sp + fu,$$

and the cost of the LSC preconditioner is

$$cost_{lsc} = 6nnz_B + 2nnz_F + 3n + 2sp + fu.$$

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

$$diff = (2nnz_F) - (2m + 2n + 2nnz_B).$$

Since we using stable elements only, $diff > 0$.

2.6. **Suitable norm to terminate the Stokes iterations.** We have noticed that when we solve the Stokes equations by SIMPLE-type preconditioned GCR method, the number of iterations depends on the viscosity (Reynolds number). See for example Table 1. Such a result is unexpected since in the Stokes equations the viscosity is only a scaling parameter and therefore the convergence should be independent of

the Reynolds number. Close inspection reveals that the accuracy of the solution is lower for high Reynolds numbers than for low ones. Hence the conclusion is that, in case of Stokes, the termination criterion should be adapted to avoid this viscosity dependence.

Grid	SIMPLER (Re=1)	SIMPLER (Re=300)
8×24	20	18
16×48	40	36
32×96	110	52

TABLE 1. Backward facing step: Solution of the Stokes problem with SILU preconditioned Bi-CGSTAB(*accuracy* of 10^{-6}).

To investigate this effect we take the SIMPLE preconditioner and solve the Stokes problem with viscosity $\nu = 1$

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}, \tag{24}$$

the convergence criteria for the outer iterations is:

$$\frac{\| \begin{bmatrix} f - Fu - B^T p \\ g - Bu \end{bmatrix} \|_2}{\| \begin{bmatrix} f \\ g \end{bmatrix} \|_2} \leq \epsilon. \tag{25}$$

Since $\nu = 1$, so the solution is obtained upto the required accuracy because no scaling is involved in the momentum and continuity equations and convergence check (28) will terminate the iterative method at the desired accuracy. However, in case of a general value of the ν we rewrite the system as:

$$\begin{bmatrix} \tilde{F} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ \tilde{p} \end{bmatrix} = \begin{bmatrix} \tilde{f} \\ g \end{bmatrix}, \tag{26}$$

where $\tilde{F} = \nu F$, $\tilde{p} = \nu p$ and $\tilde{f} = \nu f$. The SIMPLE preconditioner for (26) can be written as:

Effect of ν on the SIMPLE preconditioner

(1) Solve $\tilde{F}u^* = \tilde{f}$
(2) Solve $S\tilde{\delta p} = g - Bu^*$
(3) update $u = u^* - \tilde{D}^{-1}B^T\tilde{\delta p}$, where $\tilde{D} = \nu D$
(4) update $\tilde{p} = \tilde{\delta p}$

- Step 1 is terminated if $\frac{\|\tilde{f} - \tilde{F}u^*\|_2}{\|\tilde{f}\|_2} \leq \epsilon$, so no effect of ν on the convergence.
- In Step 2 we use $\frac{\|g - Bu^* - S\tilde{\delta p}\|_2}{\|g - Bu^*\|_2} \leq \epsilon$, so also no effect of ν on the convergence.
- For the outer iterations the usual termination criterion is

$$\frac{\| \begin{bmatrix} \tilde{f} - \tilde{F}u - B^T\tilde{p} \\ g - Bu \end{bmatrix} \|_2}{\| \begin{bmatrix} \tilde{f} \\ g \end{bmatrix} \|_2} \leq \epsilon. \tag{27}$$

We see in (26), that only the momentum equation is scaled with ν so this will effect the convergence of the outer iterative method if convergence check (27) is applied. This means that using ν in the Stokes problem effects the accuracy of the solution

of the Stokes problem. If a suitable norm is used, ν will have no effect on the convergence of the iterative method. First we shall define some quantities:

$$N_{full} = \| \begin{bmatrix} \tilde{f} - \tilde{F}u - B^T\tilde{p} \\ g - Bu \end{bmatrix} \|_2, \; N_r = \| \begin{bmatrix} \tilde{f} \\ g \end{bmatrix} \|_2 \tag{28}$$

$$N_u = \| \; \tilde{f} - \tilde{F}u - B^T\tilde{p} \; \|_2, \; N_{ru} = \| \; \tilde{f} \; \|_2 \tag{29}$$

$$N_p = \| \; g - Bu \; \|_2, \; N_{rp} = \| \; g \; \|_2 \tag{30}$$

Convergence checks: We have implemented the following options to terminate the iteration process:

(1) $N_{full} \leq \epsilon Nr$ (standard criterion)
This check shows viscosity dependence convergence in the Stokes problem. The reason is that in the overall norm, only the velocity is scaled with the viscosity.

(2) $N_u \leq \epsilon Nru$ *and* $N_p \leq \epsilon N_{rp}$
Fails to show convergence in the Navier-Stokes problem due to too small ϵN_{rp}. However, in the Stokes problem it shows viscosity independent convergence.

(3) $N_{full} \leq \epsilon Nr$ *and* $N_u \leq \epsilon N_{ru}$
This check shows viscosity independent convergence in the Stokes problem and faces no trouble in the Navier-Stokes problem. In this case, if pressure dominates the full norm, then the second condition takes care of the velocity norm to satisfy the convergence criterion.

(4) $(N_u + N_p) \leq \epsilon(N_{ru} + N_{rp})$
This convergence check shows viscosity dependence in the Stokes problem. If the pressure dominates the norm then it will show viscosity dependent convergence.

In our implementation, we tested all these conditions and option 3 is considered to be the best condition both in the Stokes and the Navier-Stokes problem since this is the only condition that encounters the effect of scaling on convergence without any problem.

3. Numerical experiments

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

(1) The Poisseuille channel flow in a square domain $(-1,1)^2$ with a parabolic inflow boundary condition and a natural outflow condition having the analytic solution: $u_x = 1 - y^2;\ u_y = 0;\ p = 2\nu x$. In case of Stokes flow $\nu = 1$.

(2) The L-shaped domain known as the backward facing step shown in Figure 1. 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.

(3) Lid driven cavity problem; flow in a square cavity $(-1,1)^2$ 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 shown in Figure 2.

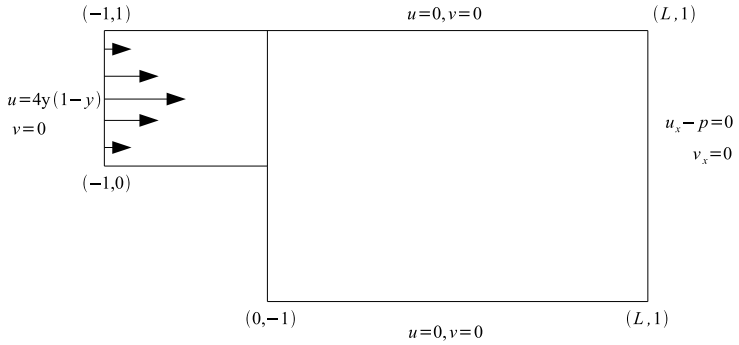

Figure 1. Backward facing step domain

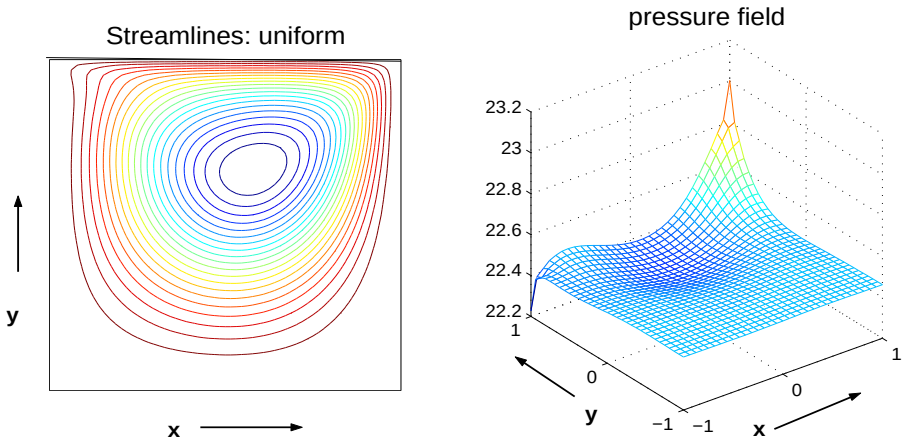

Figure 2. Streamlines and pressure field for the lid driven cavity problem

Preconditioned Krylov subspace methods are used to solve the Stokes and the Navier-Stokes problem. We divide the experiments into two sections; Section 3.1 which deals only with SIMPLE-type preconditioners and Section 3.2 which consists

of a comparison of SIMPLE-type preconditioners with the least squares commutator(LSC) preconditioner. To solve the subsystems iteratively, ILU preconditioned Krylov subspace methods are used. 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 are used: *out-iter.* is used for outer iterations, NC for no convergence, *Iter.* for iterations, inner-it-u and inner-it-p is used for iterations taken by the solver to solve subsystems in the preconditioners corresponding to velocity and pressure part, respectively. Numerical experiments are performed on an *Intel 2.66 GHz processor with 8GB RAM*. We will use IFISS package and SEPRAN for our numerical experiments. The IFISS software[1] in Matlab, can be used to solve linear systems arising from finite element discretizations of problems in computational fluid dynamics. It has built-in multigrid and Krylov subspace solvers and includes a variety of appropriate preconditioning strategies for each problem in 2D. SEPRAN[2], developed by "Ingenieursbureau SEPRA", is a general purpose finite element package used to solve a wide variety of problems in 2D and 3D programmed in FORTRAN 90. It is a powerful, modular, open ended, easy to use package for finite element analysis.

3.1. **SIMPLE-type preconditioners.** SIMPLE-type preconditioners are tested with the GCR method [26]. GCR allows variable preconditioners. A direct solver and preconditioned Bi-CGSTAB [27] as well as PCG [28] are used to solve subsystems in the preconditioners.
In report [29], we concluded that SIMPLER performs better than the SIMPLE preconditioner. However, a positive aspect of the SIMPLE preconditioner that we have observed here is that the convergence of the SIMPLE preconditioner is independent of the accuracies used to solve subsystems, while the SIMPLER preconditioner strongly depends on the inner accuracies. The larger the number of grid elements, the stronger the accuracy requirement should be for the inner solver in the SIMPLER preconditioner. In Table 2, we can see that hSIMPLER and SIMPLER depend strongly on the inner tolerance. For a large problem, SIMPLER does not give convergence even with a high accuracy for the subsystem solves. Though hSIMPLER shows better convergence than MSIMPLER, however convergence with hSIMPLER also depends on inner accuracies.

The same problem solved with SIMPLE and MSIMPLER is shown in Table 3. The inner accuracies in this case are kept constant at 10^{-1} for the velocity part and 10^{-2} for the pressure part. Both SIMPLE and MSIMPLER show convergence independent of the inner accuracies. MSIMPLER shows much faster convergence in outer iterations, inner iterations and CPU time than the other SIMPLE-type preconditioners discussed in this report.

The Navier-Stokes problem solved with various Reynolds numbers is shown in Figure 3 . We report here the number of accumulated inner iterations (Left) and the CPU time (Right). We see that MSIMPLER shows faster convergence than the other variants of SIMPLE-type preconditioners. hSIMPLER is not used as it gives only advantage in the Stokes problem.

From the discussion above, we conclude that the convergence of MSIMPLER is better than other SIMPLE-type preconditioners. Compared to other SIMPLE-type preconditioners, convergence of the MSIMPLER preconditioner is less affected by the grid size and the Reynolds number. MSIMPLER shows robust convergence behavior and the accuracies of the inner solver need not to be changed with the

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

increase in number of elements. In the next section, we will compare MSIMPLER with the least squares commutator preconditioner.

Tolerance	hSIMPLER		SIMPLER	
$(10^{p1,u1,p1})$	out-it,Time(s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it,Time(s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$
16×48				
-2 , -1 , -2	NC		NC	
-2 , -1 , -3	31, 0.6	$\frac{245}{897}$	28, 0.57	$\frac{225}{795}$
-3 , -1 , -3	22, 0.45	$\frac{170}{669}$	28, 0.58	$\frac{224}{849}$
-3 , -1 , -3				
32×96				
-2 , -1 , -2	NC		NC	
-3 , -1 , -3	NC		NC	
-3 , -1 , -4	35, 5.6	$\frac{586}{2213}$	NC	
-3 , -3 , -3	NC		NC	
-4 , -1 , -3	NC		NC	
-4 , -1 , -4	34, 5.6	$\frac{572}{2250}$	NC	
-4 , -2 , -4	41, 9.5	$\frac{1141}{2686}$	NC	
-4 , -3 , -4	34, 10.4	$\frac{1354}{2237}$	67, 20	$\frac{2680}{2654}$
64×192				
-5 , -3 , -5	45, 188	$\frac{3668}{6169}$	NC	
-4 , -3 , -5	45, 187	$\frac{3668}{5935}$	NC	
-3 , -3 , -5	46, 190	$\frac{3752}{5781}$	NC	

TABLE 2. Sensitivity of the SIMPLER preconditioner to the inner tolerances, Stokes backward facing step solved with GCR(20) with *accuracy* of 10^{-4} , PCG used as an inner solver (SEPRAN)

	SIMPLE		MSIMPLER	
Grid size	out-it,Time(s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it,Time(s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$
16×48	49, 0.8	$\frac{145}{765}$	9, 0.15	$\frac{22}{260}$
32×96	89, 8.9	$\frac{418}{2585}$	10, 0.97	$\frac{32}{568}$
64×192	193, 148	$\frac{1940}{10067}$	14, 8.4	$\frac{90}{1433}$

TABLE 3. Stokes backward facing step, GCR(20) with *accuracy* of 10^{-4} , PCG used as an inner solver (SEPRAN)

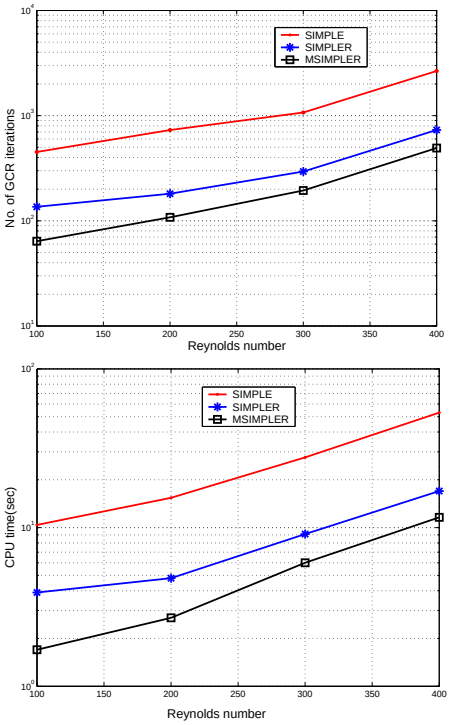

FIGURE 3. The Navier-Stokes problem solved in Q2-Q1 discretized 16×48 backward facing step with various Reynolds numbers. Number of accumulated inner iterations(Left), CPU time in seconds (Right)-(SEPRAN)

3.2. **Comparison of Preconditioners.**

3.2.1. *Comparison in 2D.* In this section, we compare the MSIMPLER preconditioner with the least squares commutator preconditioner. For outer iterations in LSC, Bi-CGSTAB(ℓ) [30] and GMRES(with modifications from Kelly) [31] are used in the literature [19]. Therefore, some of the experiments will be carried out with these two iterative methods using the IFISS package. Besides, some experiments are done with GCR(20) in SEPRAN. Subsystems in the preconditioners are solved with a direct solver, an iterative solver and a multigrid solver. In case of MG or direct solver used as inner solver, we performed the experiments within the IFISS package. We report here the total number of GMRES iterations. We keep the inner accuracy of 10^{-1} for the velocity part and 10^{-2} for the pressure part. Besides that we also used the algebraic-based preconditioner SILU [9].
Backward facing step problem: To compare the various preconditioners we first consider the solution of the Stokes and Navier-Stokes equations on the backward facing step problem.
Table 4 shows the number of iterations and CPU time required to solve the Stokes problem in the backward facing step using Q2-Q1 elements by the various preconditioners, in combination with GCR. Also the results of SILU iterations are shown. For SILU there is only one set of iterations, since there are no inner iterations.
In Table 5 we only compare LSC and MSIMPLER for the solution of the Navier-Stokes problem on the backward facing step for various values of the Reynolds number. The equations are linearized by the Picard method. The number of iterations shown is the sum of all iterations over all Picard steps. The number of Picard iterations required in LSC is sometimes larger than for MSIMPLER. For the combination LSC and RE=400, Picard does not converge. Clearly MSIMPLER behaves much better than LSC.

The difference in number of iterations between LSC and MSIMPLER decreases with increasing mesh size. This can be seen in Table 6. The system of equations for pressure and velocity part is solved by one MG cycle. The convergence is almost independent of the Reynolds number, especially for increasing mesh size. It can been seen that in most cases, the number of iterations for fixed Reynolds decreases with increase in the number of grid elements. This is due to the decrease in the cell Reynolds number (Re_c = element size/ν).
Although LSC and MSIMPLER also can be applied in case Navier-Stokes is linearized by the Newton Raphson method, Table 7 shows that its convergence is much slower than Picard linearization. The reason is that Newton may have a negative effect on the main diagonal elements of the velocity matrix. Switching from Picard to Newton linearization, the difference in number of iterations is small for MSIMPLER and large for LSC.

From our experiments we conclude that MSIMPLER performs better than LSC both in number of iterations and CPU time. This behavior is the most pronounced if we use a preconditioned Krylov subspace method as inner solver. MSIMPLER is cheaper than LSC per iteration since it does not involve the multiplication of a vector with the convection-diffusion matrix.
Lid driven cavity problem: The results for the lid driven cavity confirm our conclusions of the previous section. We have solved this problem with the IFISS package. In this case the first 8 iterations are solved by a direct solver and in the ninth Picard iteration step, the iterative solver is used. This is the default setting of IFISS. Hence the number of iterations presented in Table 8 refer to one Picard iteration only.
For increasing Reynolds number the number of iterations for both our preconditioners increases mildly. This behavior decreases for increasing mesh size. MSIMPLER

always requires less CPU-time than LSC.
Table 9 shows the effect of replacing the iterative inner solver by a direct inner solver. We compare GMRES(20) with the direct solver to see the effect on the number of outer iterations. It is clear that the number of outer iterations in case of MSIMPLER is hardly effected by using an iterative solver instead of a direct solver. In case of LSC this is no longer true. For fine grids LSC does not even converge, if an iterative inner solver is used. But even in case of a direct solver, the number of outer iterations of MSIMPLER never exceeds that of LSC. Since for a large problem a direct solver is no option. We conclude that also for this problem MSIMPLER is better than LSC.

Grid size	MSIMPLER		SILU	LSC	
	out-it,Time(s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it,Time(s)	out-it,Time(s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$
16×48	12, 0.17	$\frac{24}{346}$	87, 0.26	18, 0.37	$\frac{78}{506}$
32×96	16, 1.46	$\frac{54}{864}$	276, 3.6	24, 3.46	$\frac{201}{1318}$
64×192	23, 11.4	$\frac{145}{2308}$	1052, 56	40, 42	$\frac{674}{4049}$
128×384	32, 147	$\frac{388}{5924}$	4235, 966	69, 600	$\frac{2412}{13265}$

TABLE 4. Backward facing step: Preconditioners used in solving the Stokes problem with preconditioned GCR(20) with *accuracy* of 10^{-6} , PCG is used as an inner solver with an accuracy of 10^{-2} for the pressure and 10^{-1} for the velocity system(SEPRAN)

Grid	Re=100		Re=200		Re=400	
Grid	MSIMPLER	LSC	MSIMPLER	LSC	MSIMPLER	LSC
outer-iterations $\frac{\text{inner-it-}u}{\text{inner-it-}p}$						
16×48	$111\frac{284}{1134}$	$137\ \frac{622}{2102}$	$182\frac{581}{2017}$	$257\ \frac{1202}{3545}$	$449\frac{1697}{5235}$	$715\ \frac{3529}{8481}$
32×96	$146\frac{859}{2651}$	$238\ \frac{2332}{6735}$	$241\frac{1758}{4485}$	$472\ \frac{4383}{13955}$	$480\frac{3487}{8268}$	$820\ \frac{7772}{19752}$
64×192	$190\frac{2727}{11932}$	$393\frac{9785}{29924}$	$286\frac{4647}{16665}$	$913\ \frac{18441}{69137}$	$506\frac{8190}{27789}$	NC

TABLE 5. 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 are the accumulated iterations consumed by the outer and inner solvers (SEPRAN)

Grid	Re=100			Re=200			Re=400		
Grid	Re_c	MSIMPLER	LSC	Re_c	MSIMPLER	LSC	Re_c	MSIMPLER	LSC
outer-iterations (time in seconds)									
16×48	3.13	9(3.1)	17(5.3)	6.25	15(5.8)	27(9.2)	25	29(13.4)	73(30)
32×96	1.56	11(14.5)	16(20)	3.12	10(14.7)	15(22)	12.5	15(19)	24(28)
64×192	0.78	20(98)	24(117)	1.56	15(70)	23(106)	6.24	18(118)	22(143)

TABLE 6. Backward facing step: Solution of the Navier-Stokes problem linearized by Picard method , $Q_2 - Q_1$ discretization preconditioned Bi-CGSTAB(*accuracy* of 10^{-6}), MG is used inner solver (IFISS)

Grid	Re=100		Re=200		Re=400	
Grid	MSIMPLER	LSC	MSIMPLER	LSC	MSIMPLER	LSC
outer-iterations (time in seconds)						
16×48	11(4.5)	25(9.6)	18(8.6)	55(23.7)	44(24.7)	154(79)
32×96	13(23)	25(33)	17(33)	46(79)	31(59)	122(233)
64×192	19(140)	33(247)	22(126)	50(282)	38(235)	140(950)

TABLE 7. Backward facing step: Solution of the Navier-Stokes problem linearized by Newton method , $Q_2 - Q_1$ discretization preconditioned Bi-CGSTAB(*accuracy* of 10^{-6}), MG is used inner solver (IFISS)

Grid	Re=100		Re=500		Re=1000	
Grid	MSIMPLER	LSC	MSIMPLER	LSC	MSIMPLER	LSC
8×8	15	24	48	75	97	137
16×16	17	24	46	72	88	130
32×32	22	27	40	57	74	113
64×64	30	32	39	49	58	71

TABLE 8. Lid driven cavity: The number of outer iterations taken by preconditioned GMRES to solve the Navier-Stokes problem with accuracy 10^{-6}, using MG as inner solver (IFISS)

Grid	ILU preconditioned GMRES		Direct solver	
Grid	MSIMPLER	LSC	MSIMPLER	LSC
outer-iterations$\frac{\text{inner-it-}u}{\text{inner-it-}p}$				
Re =100				
8×8	$12\frac{43}{89}$	$17\frac{60}{198}$	2	14
16×16	$14\frac{80}{152}$	$18\frac{119}{256}$	14	16
32×32	$18\frac{182}{293}$	$29\frac{424}{1960}$	19	21
64×64	$25\frac{528}{693}$	$33\frac{1122}{3881}$	26	28
Re =500				
8×8	$32\frac{151}{236}$	$36\frac{227}{267}$	30	36
16×16	$34\frac{180}{357}$	$40\frac{329}{466}$	33	36
32×32	$28\frac{289}{454}$	$38\frac{714}{1423}$	30	33
64×64	$35\frac{981}{1037}$	NC	35	38
Re =1000				
8×8	$70\frac{346}{519}$	$73\frac{478}{524}$	57	69
16×16	$68\frac{414}{724}$	$78\frac{717}{791}$	71	80
32×32	$60\frac{616}{935}$	$86\frac{1633}{1700}$	55	60
64×64	$47\frac{1262}{1253}$	NC	45	53

TABLE 9. Lid driven cavity: Preconditioned GCR(20) is used to solve the Navier-Stokes problem with accuracy 10^{-6} (IFISS)

Effect of stretching. To investigate the effect of stretching on the preconditioners we consider the following two types of stretching.

Length stretching: This is the most simple type of stretching. The length of the channel is increased in the flow direction keeping the number of elements constant. Hence the grid remains equidistant in both directions, but the aspect ratio of the elements increases.

Elastic stretching: Elements sizes are taken small in regions where there is a large change in the solution and large when the change is small. Hence the mesh is no longer equidistant. Instead element sizes become larger with some factor. Both ways of stretching are shown in Figure 4.
Another type of stretching: increasing the number of elements in a direction perpendicular to the flow will not be discussed in this report. In this case, the aspect ratio of the elements in the grid remains the same.

Table 10 shows the convergence of the iterative solver on a grid with length stretching. It concerns a Stokes channel flow. From the table one may conclude that SIMPLER is very sensitive to length stretching. In case of high aspect ratio, convergence is not guaranteed. The other preconditioners have no problem at all with high aspect ratios. Since hSIMPLER reduces to SIMPLER in case of Navier-Stokes it can also not used in case of length stretching.

The story is completely different in case of elastic stretching. As long as the grid size is constant per direction, MSIMPLER and LSC perform well. However, if the grid size changes, the number of iterations increases considerably even if the increase in length between adjacent elements is small.
Table 11 shows the results for a square cavity and flow over a plate. The stretch factor is the ratio of the length of two adjacent elements. The IFISS package is used and each inner solve is performed by a direct solver to eliminate the effect of inaccurate inner solvers.
It is clear that scaling with the diagonal of the velocity mass matrix (MSIMPLER, LSC) is in general less effective than the scaling with the diagonal of the velocity matrix (SIMPLER, LSC($diag(F)$) as soon the stretching factor increases. These methods are only superior for equidistant grids. If we use an iterative solver, SIMPLER even does not converge in some cases. So we may conclude that element stretching introduces large problems for our iterative methods. Probably a better scaling matrix is required in those cases.

Length	SIMPLE	SIMPLER	hSIMPLER	MSIMPLER	LSC
	Iter.	Iter.	Iter.	Iter.	Iter.
L=10	70	51	36	18	21
L=50	94	NC	45	21	20
L=100	75	105	37	16	16
L=200	56	NC	26	13	16

TABLE 10. Solution of the channel Stokes problem, $16\times16\ Q_2-Q_1$ discretization, preconditioned GCR (*accuracy* of 10^{-6}) and subsystem solved with PCG(SEPRAN)

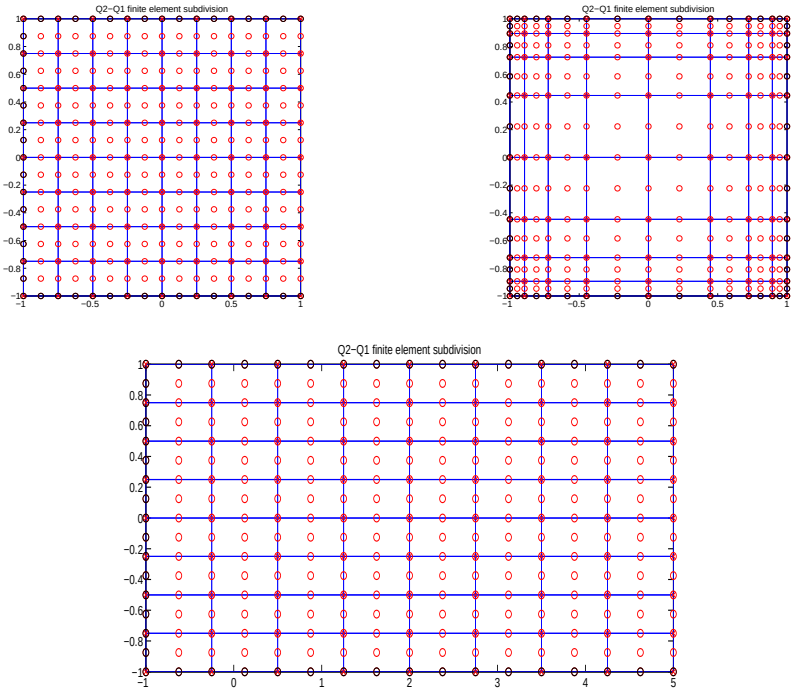

FIGURE 4. Q2-Q1 stretched grids, normal grid (Left), Elastic stretching (Right), Length stretching (Below).

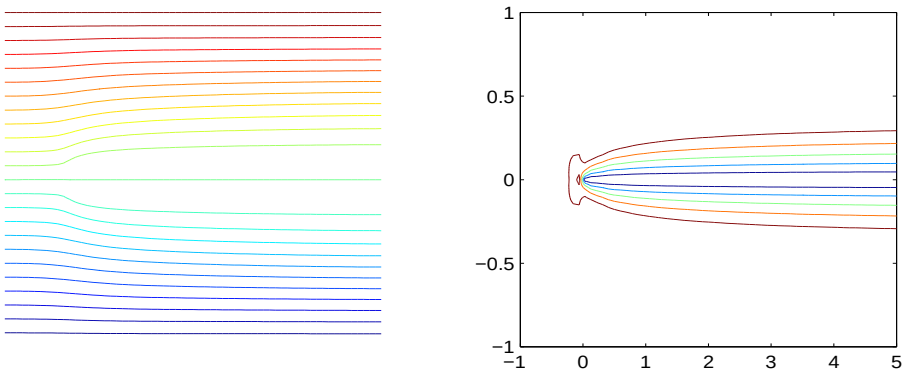

FIGURE 5. Flow over a plate: Streamline plot (Left) and horizontal velocity contour plot (Right) of a 64×64 grid with $\nu = 1/100$.

Grid	stretch-factor	SIMPLER	MSIMPLER	LSC	LSC($diag(F)$)
		Iter.(sec)	Iter.(sec)	Iter.(sec)	Iter. (sec)
Re=200, Cavity flow					
16×16	1	22(0.7)	19(0.6)	22(0.6)	26(0.76)
16×16	1.1669	29(0.82)	31(0.88)	34(0.92)	33(0.90)
32×32	1	34(5.3)	21(2.7)	24(2.9)	35(5.1)
	1.097	49(7.73)	51(8)	61(9)	54(8)
64×64	1	71(58)	30(20)	33(21)	60(46)
	1.056	NC	95(81)	124(102)	99(81)
Re=100, Flow over a plate					
32×32	1	85(11.31)	42(5.3)	44(5.2)	76(9.3)
	1.05	105(14)	66(8.5)	67(8.1)	104(13)
	1.1	123(16)	128(17)	120(15)	126(16)
	1.15	129(17.6)	233(34)	222(30.5)	137(17.6)
Re=200, Flow over a plate					
32×32	1.1	161(22.5)	159(22)	157(20.4)	166(21.8)
	1.15	177(25)	305(47)	306(45)	177(23.6)

TABLE 11. Solution of the Navier-Stokes problem, $Q_2 - Q_1$ discretization, preconditioned GCR (*accuracy* of 10^{-6}) and subsystem solved with direct solver for elastic stretching(IFISS)

3.2.2. *Experiments in 3D.* To investigate the performance of the iterative methods in 3D we have solved the 3D backward facing step problem. We have limited ourselves to non-stretched grids. Besides the iterative solvers derived in this report we also use the SILU iterative solver for comparison. Mark that iterations in SILU are noted as outer iterations, although there are no inner iterations for that method. Table 12 shows the results for the Stokes problem. We see that both SIMPLER and LSC fail for increasing mesh size. MSIMPLER is always cheaper than SILU especially for increasing grid size where the gain becomes large. The accuracy required is 10^{-6}.

Table 13 gives the results for Navier-Stokes problem linearized by the Picard method. In each non-linear iteration we use an accuracy of 10^{-2} which is also sufficient to let LSC converge. In this case the difference between MSIMPLER and SILU is much smaller. LSC converges but takes approximately twice the time needed for the other methods in case of fine grids. If we replace the hexahedrons by tetrahedrons the results for the Navier-Stokes are completely different as shown in Table 14. Now SIMPLE converges much faster than MSIMPLER and LSC, and SILU is superior to the other methods. The diagonal of the velocity mass matrix does not seem a good scaling matrix in this case.

Grid size	SIMPLE		MSIMPLER		SILU	LSC	
	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it(t_s)	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$
$8 \times 8 \times 16$	50(4.4)	$\frac{100}{923}$	14(1.3)	$\frac{27}{514}$	149(2.6)	21(2.3)	$\frac{50}{756}$
$16 \times 16 \times 32$	95(113)	$\frac{328}{3291}$	20(25)	$\frac{61}{1313}$	557(77)	29(50)	$\frac{148}{1913}$
$24 \times 24 \times 48$	130(617)	$\frac{439}{6518}$	19(94)	$\frac{51}{1833}$	698(373)	NC	
$32 \times 32 \times 40$	NC		26(208)	$\frac{93}{3069}$	978(720)	NC	

TABLE 12. 3D Backward facing step: Preconditioners used in the Stokes problem with preconditioned GCR(20) with *accuracy* of 10^{-6} (SEPRAN) using Q2-Q1 hexahedrons

Grid size	MSIMPLER		SILU	LSC	
	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it(t_s)	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$
$8 \times 8 \times 16 (Re = 100)$	63(11)	$\frac{135}{1026}$	207(8.0)	84(15)	$\frac{267}{1948}$
$8 \times 8 \times 16 (Re = 200)$	90(14.3)	$\frac{166}{1725}$	311(11.5)	132(20)	$\frac{337}{2830}$
$16 \times 16 \times 32 (Re = 400)$	153(269)	$\frac{485}{4757}$	901(207.6)	271(553)	$\frac{1300}{10327}$
$24 \times 24 \times 48 (Re = 400)$	154(1187)	$\frac{675}{6498}$	2072(1400)	297(2800)	$\frac{1961}{18056}$
$32 \times 32 \times 40 (Re = 400)$	209(2479)	$\frac{1322}{10249}$	3008(2607)	387(4600)	$\frac{3970}{29190}$

TABLE 13. 3D Backward facing step: Preconditioners used in solving the Navier-Stokes problem with preconditioned GCR(20) with *accuracy* of 10^{-2} (SEPRAN) using Q2-Q1 hexahedrons

Re	SIMPLE		MSIMPLER		SILU	LSC	
	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$	out-it(t_s)	out-it(t_s)	$\frac{\text{inner-it-}u}{\text{inner-it-}p}$
50	107(255)	$\frac{498}{107}$	128(309)	$\frac{597}{256}$	210(105)	118(259)	$\frac{405}{236}$
200	203(551)	$\frac{1232}{203}$	248(698)	$\frac{1549}{496}$	392(188)	275(753)	$\frac{1609}{550}$
400	238(716)	$\frac{1700}{238}$	272(1359)	$\frac{1932}{544}$	547(175)	389(1676)	$\frac{3050}{778}$

TABLE 14. 3D Backward facing step: Preconditioners used in the Navier-Stokes problem with preconditioned GCR(20) with *accuracy* of 10^{-2} (SEPRAN) using $16 \times 16 \times 32$ Q2-Q1 tetrahedron, the accumulated number of iterations are reported in this table

Conclusions

In this report we have studied the convergence behavior of some block preconditioner for Stokes and Navier-Stokes both in 2D and 3D. Results for various grid sizes and Reynolds numbers have been investigated. In some cases we also compared the convergence with an algebraic preconditioner (SILU). Unfortunately there is no unique conclusion, since the results depend on stretching and the type of elements used. We come to the following observations:

- MSIMPLER is at present the fastest of all SIMPLE-type preconditioners, except in the case of 3D tetrahedrons, where its convergence becomes poor.
- 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 and therefore can 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 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 difference are observed. It appears that the combination of LSC with MG is nice but LSC with PCG inner solver can take many iterations and much CPU time.
- In our experiments, MSIMPLER proved to be cheaper than SILU except in case of 3D tetrahedrons.
- MSIMPLER has no problem with length stretching, whereas some of the other preconditioners have. However in case of elastic stretching, the convergence of MSIMPLER and LSC becomes poor.

We observed that, for the grid having elements with small aspect ratio, the convergence of MSIMPLER is encouraging for the rectangular elements in 2D and hexahderons elements in 3D. However, the nice convergence properties of MSIMPLER fade when it is applied to the grid with trianglular elements in 2D and tetrahedrons in 3D. MSIMPLER shows also problems in stretched grids. These are the main points of our further research.

References

[1] 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.

[2] 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.

[3] J. A. Meijerink and H. A. van der Vorst. An Iterative Solution Method for Linear Systems of Which the Coefficient Matrix is a Symmetric M-Matrix. *Math. of Comp.*, 31(137):148–162, January 1977.

[4] I. S. Duff and G. A. Meurant. The effect of ordering on preconditioned conjugate gradients. *BIT*, 29(4):635–657, 1989.

[5] L. C. Dutto. The effect of ordering on preconditioned GMRES algorithm, for solving the compressible Navier-Stokes equations. *Int. J. Numer. Meth. Engng.*, 36(3):457–497, 1993.

[6] M.Benzi, D. B. Szyld, and A. van Duin. Orderings for Incomplete Factorization Preconditioning of Nonsymmetric Problems. *SIAM J. Sci. Comput.*, 20(5):1652–1670, 1999.

[7] Michele Benzi and Miroslav Tuma. A Sparse Approximate Inverse Preconditioner for Nonsymmetric Linear Systems. *SIAM Journal on Scientific Computing*, 19(3):968–994, 1998.

[8] Matthias Bollhofer and Yousef Saad. Multilevel Preconditioners Constructed From Inverse-Based ILUs. *SIAM J. Sci. Comput.*, 27(5):1627–1650, January 2006.

[9] M. ur Rehman, C. Vuik, and G. Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. *International Journal for Numerical Methods in Fluids*, Electronic print available (Early view), 2007.

[10] M. Benzi and M. A. Olshanskii. An Augmented Lagrangian-Based Approach to the Oseen Problem. *SIAM J. Sci. Comput.*, 28(6):2095–2113, 2006.
[11] 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.
[12] Alain Gauthier, Fausto Saleri, and Alessandro Veneziani. A fast preconditioner for the incompressible Navier Stokes Equations. *Comput. Vis. Sci.*, 6(2):105–112, 2004.
[13] D. Kay, D. Loghin, and A. Wathen. A Preconditioner for the Steady-State Navier-Stokes Equations. *SIAM J. Sci. Comput.*, 24(1):237–256, 2002.
[14] A. C. de Niet and F. W. Wubs. Two preconditioners for saddle point problems in fluid flows. *Int. J. Numer. Meth. Fluids*, 54(4):355–377, 2007.
[15] M. A. Olshanskii. An Iterative Solver for the Oseen Problem and Numerical Solution of Incompressible Navier-Stokes Equations. *Numer. Linear Algebra Appl.*, 6:353–378, 1999.
[16] 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.
[17] M. Benzi, G. H. Golub, and J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 14:1–137, 2005.
[18] M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *J. Comput. Phys.*, 182(2):418–477, 2002.
[19] H. C. Elman, D. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers with applications in incompressible fluids dynamics*. Oxford University Press, Oxford, 2005.
[20] P. Wesseling. *Principles of computational fluid dynamics*, volume 29. Springer Series in Computational Mathematics, Springer, Heidelberg, 2001.
[21] S. V. Patankar. *Numerical heat transfer and fluid flow*. McGraw-Hill, New York, 1980.
[22] C. Vuik and A. Saghir. The Krylov accelerated SIMPLE(R) method for incompressible flow. Report 02-01, Delft University of Technology, Department of Applied Mathematical Analysis, Delft, 2002.
[23] J. Blair Perot. An analysis of the fractional step method. *J. Comput. Phys.*, 108(1):51–58, 1993.
[24] J. Blasco, R. Codina, and A. Huerta. A fractional-step method for the incompressible Navier-Stokes equations related to a predictor-multicorrector algorithm. *International Journal for Numerical Methods in Fluids*, 28(10):1391–1419, 1998.
[25] Alexandre Joel Chorin. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, 22(104):745–762, 1968.
[26] 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.
[27] 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.
[28] 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.
[29] M. ur Rehman, C. Vuik, and G. Segal. Preconditioners for the incompressible Navier-Stokes equations. Technical report, TU Delft, 2007.
[30] G.L.G. Sleijpen and D.R. Fokkema. BiCGstab(ell) for Linear Equations involving Unsymmetric Matrices with Complex Spectrum ,. *ETNA*, 1:11–32, 1993.
[31] T. Kelly. *Iterative methods for linear and nonlinear equations*. SIAM, Philadelphia, 1995.

Appendix A. SIMPLER formulations

We have seen that SIMPLER preconditioner /algorithm can be written in form of Symmetric Block Gauss-Seidal [22]. This form contains two Poisson solves as well as two velocity solves. The classical form of the SIMPLER method contains only one velocity solve . We shall show that the second velocity solve has no effect on the convergence of SIMPLER. The classical from of SIMPLER is:

SIMPLER algorithm with one velocity solve:

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

where velocity u^k is estimated from prior iterations. r_u and r_p are the residuals in the velocity part and pressure part, respectively. D is the diagonal of the convection diffusion matrix and $\hat{S} = -BD^{-1}B^T$, an approximation to the Schur complement. The Symmetric Block Gauss-Seidal form of the SIMPLER preconditioner can be written as:

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

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix} \right), \tag{A-2}$$

where u^k and p^k in (16) are obtained from the previous step (both zero in our case) and

$$A = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \tag{A-3}$$

$$B_R = \begin{pmatrix} I & -D^{-1}B^T \\ 0 & I \end{pmatrix}, \; M_R = \begin{pmatrix} F & 0 \\ B & \hat{S} \end{pmatrix} \text{ and} \tag{A-4}$$

$$B_L = \begin{pmatrix} I & 0 \\ -BD^{-1} & I \end{pmatrix}, \; M_L = \begin{pmatrix} F & B^T \\ 0 & \hat{S} \end{pmatrix}. \tag{A-5}$$

The steps given in (A-1) and (A-2) contain two Poisson solves, two velocity subproblems solves- posed to one velocity solve in the classical algorithm- and matrix vector updates.

A.1. **The SIMPLER preconditioner with one velocity solve.** In the SIMPLER preconditioner, we use one step of the SIMPLER method. Therefore, we initialize u^k and p^k to zero in (A-1). This reduces (A-1) to

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = M_L^{-1} B_L \begin{pmatrix} r_u \\ r_p \end{pmatrix}, \tag{A-6}$$

Rewriting (A-6) leads to

$$p^* = \hat{S}^{-1}(r_p - BD^{-1}r_u), \tag{A-7}$$

and

$$u^* = F^{-1}(r_u - B^T p^*), \tag{A-8}$$

assuming we use exact inverses. Next we consider (A-2). First compute the residual part,

$$\begin{pmatrix} r_{un} \\ r_{pn} \end{pmatrix} = \begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix}. \tag{A-9}$$

The velocity part becomes:

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

If we substitute u^* from (A-8) in r_{un} we get

$r_{un} = r_u - FF^{-1}(r_u - B^T p^*) - B^T p^* = 0$. The pressure part is equal to:

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

therefore, (A-5) reduces to

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \begin{pmatrix} 0 \\ r_{pn} \end{pmatrix}. \tag{A-10}$$

The formulation (A-6) and (A-10) will give rise to three steps in the SIMPLER preconditioner because there is no need to perform an extra velocity solve in (A-10) when the right-hand side is zero.

$$\delta p = \hat{S}^{-1}(r_p - Bu^*), \tag{A-11}$$

$$u^{k+1} = u^* + BD^{-1}\delta p, \tag{A-12}$$

and

$$p^{k+1} = p^* + \delta p. \tag{A-13}$$

In the next section, we will show that this also holds in the SIMPLER algorithm.

A.2. **The SIMPLER algorithm with one velocity solve.** The SIMPLER algorithm in the Symmetric Block Gauss-Seidal form is given:

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} u^k \\ p^k \end{pmatrix} + M_L^{-1} B_L \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^k \\ p^k \end{pmatrix} \right), \tag{A-14}$$

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + B_R M_R^{-1} \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix} \right), \tag{A-15}$$

where A represents the complete matrix given in (4), u^k and p^k in (16) are obtained from the previous step

$$B_R = \begin{pmatrix} I & -D^{-1}B^T \\ 0 & I \end{pmatrix}, \; M_R = \begin{pmatrix} F & 0 \\ B & \hat{S} \end{pmatrix} \text{ and} \tag{A-16}$$

In this case we assume that u^k and p^k are non-zero. We are interested in computing the residual in (A-15) that is:

$$\begin{pmatrix} r_{un} \\ r_{pn} \end{pmatrix} = \begin{pmatrix} r_u \\ r_p \end{pmatrix} - A \begin{pmatrix} u^* \\ p^* \end{pmatrix}, \tag{A-17}$$

From (A-14), we can write

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

First we evaluate AM_L^{-1},

$$AM_L^{-1} = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} F^{-1} & -F^{-1}B^T\hat{S}^{-1} \\ 0 & \hat{S}^{-1} \end{pmatrix} = \begin{pmatrix} I & 0 \\ BF^{-1} & -BF^{-1}B^T\hat{S}^{-1} \end{pmatrix}. \tag{A-19}$$

And next $AM_L^{-1}B_L$,

$$AM_L^{-1}BL = \begin{pmatrix} I & 0 \\ BF^{-1} & -BF^{-1}B^T\hat{S}^{-1} \end{pmatrix} \begin{pmatrix} I & 0 \\ -BD^{-1} & I \end{pmatrix} = \begin{pmatrix} I & 0 \\ B_r & S_r \end{pmatrix}, \tag{A-20}$$

where $B_r = BF^{-1} + BF^{-1}B^T\hat{S}^{-1}BD^{-1}$ and $S_r = -BF^{-1}B^T\hat{S}^{-1}$. (A-18) becomes

$$A \begin{pmatrix} u^* \\ p^* \end{pmatrix} = A \begin{pmatrix} u^k \\ p^k \end{pmatrix} + \begin{pmatrix} I & 0 \\ B_r & S_r \end{pmatrix} \begin{pmatrix} r_u - Fu^k - B^T p^k \\ r_p - Bu^k \end{pmatrix}. \tag{A-21}$$

$$A\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} Fu^k + B^T p^k \\ Bu^k \end{pmatrix} + \begin{pmatrix} r_u - Fu^k - B^T p^k \\ B_r(r_u - Fu^k - B^T p^k) + S_r(r_p - Bu^k) \end{pmatrix}, \quad \text{(A-22)}$$

$$A\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} r_u \\ Bu^k + B_r(r_u - Fu^k - B^T p^k) + S_r(r_p - Bu^k) \end{pmatrix}. \quad \text{(A-23)}$$

Updating (A-17) by using (A-23) leads to

$$\begin{pmatrix} r_{un} \\ r_{pn} \end{pmatrix} = \begin{pmatrix} 0 \\ r_p - (Bu^k + B_r(r_u - Fu^k - B^T p^k) + S_r(r_p - Bu^k)) \end{pmatrix}. \quad \text{(A-24)}$$

So again we see that the right-hand side for the velocity solve in (A-17) is zero. In the proofs we used exact solvers. We have done some experiments with inexact solvers and found no major difference between the number of iterations with one or two velocity solves. For this reason we only use the one velocity solve formulation.