DELFT UNIVERSITY OF TECHNOLOGY

REPORT 07-15

Preconditioners for the incompressible Navier-Stokes equations

M. ur Rehman, C. Vuik and G. Segal

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2007

PRECONDITIONERS FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS

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

Abstract. We compare a number of efficient preconditioners, published in recent papers, with our own developed SILU method. Some preconditioners are modified in order to get a more efficient implementation. All preconditioners are applied to a couple of standard benchmark problems. The final goal is to select the best preconditioners for large three - dimensional engineering problems.

1. Introduction

The Navier-Stokes equations describe the general motion of fluid. Analytical solution of these equations is nearly impossible, so a numerical discretization technique is necessary. In our case we apply the finite element method (FEM) to discretize the incompressible Navier-Stokes equations. The resulting system of non-linear equations has to be linearized before using a linear solver. Both Newton and Picard methods could be applied for the linearization. The resulting systems of linear equations gives rise to a "saddle - point" problem, with a large number of zeros on the main diagonal. Efficient solution of these equations by an iterative solver is only possible in combination with a suitable preconditioner. In ur Rehman et al [17], we compared SILU (Saddle point ILU preconditioner) with some block preconditioners. In general, [17] is more focused on the introduction to the new ILU method. In this report, our goal is to compare a number of block - preconditioners, published in recent literature, with the straight-forwards S ILU preconditioner, suitable for saddle point problems. The comparison has been done on the basis of two benchmark problems: the simple channel flow and the two dimensional backward facing step. Based on this report, we will make a choice of the most favorite preconditioner, and implement them in the FEM package SEPRAN. This allows us to compare the methods for complex 3D problems.

2. Incompressible Navier-Stokes equations

The incompressible Navier-Stokes equations, given as

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

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

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

$$\mathbf{u} = \mathbf{w} \text{ } on \text{ } \partial\Omega_D, \ \nu\frac{\partial\mathbf{u}}{\partial\mathbf{n}} - \mathbf{n}p = 0 \text{ on } \partial\Omega_N, \text{ where } \mathbf{w} \text{ is a given function.}$$

The system given in (1) and (2) is discretized by the finite element method. For the unique solution of $\mathbf{u}$ and p, the elements used in the discretization must satisfy the well known LBB(inf-sup) condition [1], [2], [3]. Elements that satisfy LBB conditions are known as stable elements. Two well-known families of stable elements are distinguished: Taylor Hood (continuous pressure approximation) and Crouzeix Raviart (discontinuous pressure) [5] [6]. Elements that do not satisfy the LBB condition are stabilized and commonly known as stabilized elements. In this report, we only use stable elements. Due to the presence of the convective term $(\mathbf{u} \cdot \nabla \mathbf{u})$ in the momentum equation, the discretization of the Navier-Stokes equation leads to a system of non-linear equations. Next the Navier-Stokes system are linearized by a Picard or Newton method. The linearized Navier-Stokes equations can be written in matrix notation as:

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

where F is the convection diffusion operator, B^T is the gradient operator, and B is the divergence operator. In order to solve the linear system (3) we will apply a Krylov subspace method. To that end we use GMRES [9], GCR [16] and Bi-CGSTAB [10] with a suitable preconditioner. Before discussing the preconditioners, we will also consider a slightly adapted system; where we replace the matrix F by $F+\gamma B^T W^{-1} B$, with γ a parameter and W a suitable positive definite matrix, only used for scaling purposes. Due to the incompressibility constraint, system (3) is equivalent with

$$\begin{bmatrix} F_\gamma & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{4}$$

where $F_\gamma = F+\gamma B^T W^{-1} B$. This adapted equation is known as the augmented Lagrangian formulation. It is closely related to the classical Uzawa iteration scheme [7] and also to the penalty function method [8]. An important point with respect to the augmented matrix F_γ is that the structure of this matrix may be different from the matrix F.

Definition 1:
We define A_1 as the non-zero pattern of the matrix F and A_2 as the non-zero pattern of the matrix F_γ.

Figure 1 shows the non-zero pattern (A_1) of the original matrix and of the updated matrix F_γ with diagonal matrix W (A_2). A_1 results in a small profile. In the A_2 non-zero pattern we see a coupling of velocity components due to the product $B^T B$. In case of Crouzeix Raviart elements the extra coupling is not a big problem since in practice velocity components are always coupled if the viscosity is not a constant. However, in case of Taylor-Hood elements, the non-zero pattern A_2 contains much more elements then pattern A_1, even if the velocity components are coupled.

In the next section we consider various preconditioners that are used to solve either system (3) or (4).

3. Block preconditioners for the incompressible Navier-Stokes problem

In general, preconditioners are used to improve the convergence of Krylov solvers. Instead of solving a system $\mathcal{A}x = b$, one solves a system $P^{-1}\mathcal{A}x = P^{-1}b$, with P the preconditioner. A good preconditioner must satisfy at least the following properties.

- The application of the preconditioner must improve the convergence behavior. Usually this implies that the spectrum of $P^{-1}\mathcal{A}$ must be more favorable than that of A. Such a spectrum must not be close to zero, and perfectly clustered around one.
- Solution of the equation of the form $Pz = r$ must be simple.

- Construction of the preconditioner should be efficient in both CPU time and memory.

For an overview of preconditioners, we refer to [25], [19] and [29].

We especially discuss preconditioners, that are developed for saddle point problems. Our goal is to get preconditioners with the property, that the number of iterations in the linear solver is nearly independent of the Reynolds number. Besides that, grid refinement should only give a small increase in the number of iterations. We shall distinguish between two types of preconditioners,

(1) Block preconitioners, which have the property that velocity and pressure solver are decoupled during iterations.
(2) ILU preconditioners, that are applied to the integrated system. These type of preconditioners require at least some form of pivoting.

In this section, we will focus on the block preconditioners. Block preconditioners, exploit the block structure of the Navier-Stokes problem. One easily verifies that the block LDU decomposition of (3) can be written as

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

where $S = -BF^{-1}B^T$, is known as the Schur complement matrix. Combination of L, D and U leads to three types of preconditioners.

$$\begin{aligned}
D^{-1} &= \begin{bmatrix} F^{-1} & 0 \\ 0 & S^{-1} \end{bmatrix} && \text{Block diagonal preconditioner, (a)} \\
[(DU)]^{-1} &= \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix}^{-1} && \text{Block triangular preconditioner, (b)} \\
\begin{bmatrix} (LD)U \end{bmatrix}^{-1} &= \begin{bmatrix} I & \tilde{F}^{-1}B^T \\ 0 & I \end{bmatrix}^{-1} \begin{bmatrix} \hat{F} & 0 \\ B & \hat{S} \end{bmatrix}^{-1} && \text{SIMPLE preconditioner [22], (c),}
\end{aligned} \tag{6}$$

where $\tilde{F} = diag(F)$, $\hat{S} = B\tilde{F}^{-1}B^T$, $S = BF^{-1}B^T$ and $\hat{F}^{-1}$ is an approximation of F^{-1}. $\hat{F}^{-1}v$ may be, for example the result of some iterations of an iterative solver.

It is clear, from formulation (6-a), that the block diagonal preconditioner separates the computation of velocity and pressure. It can be shown, that the combination of GMRES, with this preconditioner, converges in at most three iterations to the exact solution, provided the matrices F^{-1} and S^{-1} in (6-a) are computed exactly [28]. In general the use of the exact inverse of these matrices is not very practical. In order to get a cheap solution, both F^{-1} and S^{-1} have to be approximated. Of course by doing so, the number of GMRES iterations, increases. In case of a Stokes problem, this preconditioner is symmetric, positive definite. Usually, the Schur complement matrix is approximated by a simplified matrix.

The block triangular preconditioner (6-b) also separates pressure and velocity computations. The extra cost per iteration, is the multiplication of the matrix B^T by a vector, which is almost negligible. Since according to [12], the preconditioner (6-b) requires as half as many iterations as (6-a), in practice (6-b) is preferred. Eigenvalue analysis of preconditioner (6-b) suggests that GMRES converges in two iterations, if exact arithmetic is used [28].

The iteration scheme for the block triangular preconditioner (6-b) can be written as

- $r = [r_v,\ r_p]$ and $z = [z_v,\ z_p]$
- Solve $Sz_p = r_p$
- Update $r_v = r_v - B^T z_p$
- Solve $Fz_v = r_v$

The SIMPLE preconditioner (6-c) is a different type of preconditioner. Theoretical results shows that if $\tilde{F} = F$, SIMPLE preconditioned GMRES converges in one iteration [23]. We shall compare its practical performance with that of the other preconditioners.

In all the preconditioners (6), the expensive part, is the computation of F^{-1} and S^{-1}. In practice both matrices are approximated . The type of approximation, defines the actual preconditioner.

In the next section, we shall treat some examples of such preconditioners.

4. Examples of preconditioners for the Navier-Stokes equations

In this section we shall treat some examples of popular block preconditioners. We start with diagonal block preconditioners, followed by block triangular preconditioners and SIMPLE. Finally we treat the coupled ILU-type preconditioners.

4.1. The grid-div preconditioner.

A special block diagonal preconditioner known as grid-div preconditioner, is given by

$$P_{GD} = \begin{bmatrix} F + \gamma BB^T & 0 \\ 0 & I/\gamma \end{bmatrix}, \tag{7}$$

where γ is a scaling parameter [30]. The preconditioner itself is of augmented type, but it is applied to the original system (3). Eigenvalue analysis of this preconditioner shows, that the preconditioned system has n eigenvalues 1, with n the number of velocity unknowns. The remaining eigenvalues are equal to $-\frac{\gamma\mu_i}{1+\gamma\mu_i}$, hence depends on γ. μ_i are the eigenvalues of the Schur complement matrix. For the Stokes problem, this preconditioner is symmetric positive definite if $\gamma \geq 0$. In practice $(F + \gamma BB^T)^{-1}v$ is solved by an iterative solver.

4.2. Augmented Lagrangian Approach(AL).

An example of a block triangular preconditioner is the AL preconditioner of Benzi and Olshanskii [27]. This preconditioner is based on and applied to the augmented system (4). The inverse Schur complement is approximated by

$$\hat{S}^{-1} = -(\nu\hat{Q}_p^{-1} + \gamma W^{-1}), \tag{8}$$

where $\hat{Q}_p$ denotes the approximate pressure mass matrix, and ν is the viscosity. Usually W is also replaced by $\hat{Q}_p$ and γ is the parameter in the augmented system (4).

In case of a constant pressure approximation, $\hat{Q}_p$ is already diagonal. In case of a linear pressure approximation, $\hat{Q}_p$ is constructed as a diagonal matrix, which is spectrally equivalent to Q_p [27]. Hence the computation of $\hat{S}^{-1}$ is very cheap. Results in [27] reveal that convergence of an iterative method with this preconditioner is independent of the mesh size and Reynolds number. Note that this concerns only the number of outer iterations. The number of inner iterations for the solution of $(F + \gamma BB^T)v = f$, may depend on the grid size and Reynolds number. A good choice of γ is important. Usually, the parameter γ is taken equal to 1.

In [27], multigrid iterations are recommended for the solve step $(F+\gamma BB^T)z_v = r_v$. In this report, however, we shall use an ILU preconditioned GMRES [9] or Bi-CGSTAB [10] method.

4.3. **Least Squares Commutator (LSC).** A different approximation of the Schur complement in the block triangular preconditioner is the LSC preconditioner of Elman, Howle, Shadid, Shuttleworth and Tuminaro [15]. This method is applied to the original system of equations (3). The approximation is defined by the complex expression:

$$BF^{-1}B^T \approx (BQ^{-1}B^T)(BQ^{-1}FQ^{-1}B^T)^{-1}(BQ^{-1}B^T), \tag{9}$$

where Q is the velocity mass matrix. Since the inverse of the matrix Q^{-1} is dense, Q is usually replaced with its diagonal $\hat{Q}$. This preconditioner only converges for stable elements. Each iteration requires two solve steps of Poisson-type matrices. According to [29] LSC is independent of the mesh size and mildly dependent on the Reynolds number.

4.4. **Artificial compressibility (AC) preconditioner.** A slightly different type of preconditioner is the AC preconditioner [30]. This preconditioner is based on the original system (3), where the zero pressure matrix is replaced by the diagonal matrix $-I/\gamma$:

$$P_{AC} = \begin{bmatrix} F & B^T \\ B & -I/\gamma \end{bmatrix}. \tag{10}$$

This is equivalent to adding some artificial compressibility to the continuity equation. In this case only the preconditioner is adapted. The original system (3) is solved.

This preconditioned system, also has n eigenvalues 1. The remaining eigenvalues are equal to $\frac{\gamma\mu_i}{1+\gamma\mu_i}$. So apart from the minus sign the eigenvalues of GD and AC are the same. The P_{AC} preconditioner is written as LDU in the following way

$$\begin{pmatrix} F & B^T \\ B & -I/\gamma \end{pmatrix}^{-1} = \begin{pmatrix} I & 0 \\ \gamma B & I \end{pmatrix} \begin{pmatrix} (F+\gamma B^T B)^{-1} & 0 \\ 0 & (-I/\gamma)^{-1} \end{pmatrix} \begin{pmatrix} I & \gamma B^T \\ 0 & I \end{pmatrix}. \tag{11}$$

From this expression it is clear that there is some relation with the GD preconditioner. Of course the extra term, $\gamma B^T B$, will introduce extra non-zero entries in the matrix F_γ. The convergence of the subsystem, $F_\gamma z_v = r_v$, strongly depends on γ. The larger γ, the more difficult it is to solve the system. This is similar to the penalty function method. Although for this type of system multigrid iterations are suggested, we apply an ILU preconditioned Krylov method.

4.5. **SIMPLE type preconditioners.** A complete different class of block preconditioners is formed by the SIMPLE family. SIMPLE (Semi Implicit Method for Pressure Linked Equations) [20], is a classical algorithm for solving the Navier-Stokes equations, discretized by a finite volume technique. However, SIMPLE can also be considered as a distributive method, in which a system $Ax = b$ is postconditioned by a matrix B, such that $ABy = b,\ x = By$ is easy to solve [21].

The SIMPLE algorithm reads:

(1) Estimate pressure p^* (From prior iteration)
(2) Solve $Fu^* = r_v - B^T p^*$
(3) Solve $S\delta p = r_p - Bu^*$, where $S = -BD^{-1}B^T$, $D = diag(F)$
(4) update u and p,
    $u = u^* - D^{-1}B^T\delta p$ and $p = p^* + \delta p$
(5) If not converged, $p^* = p$ and goto 1.

The initial p^* equal to zero, the SIMPLE method can also be written in distributive iterative method form [22].

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^k \\ p^k \end{pmatrix} + P \left(\begin{pmatrix} f \\ 0 \end{pmatrix} - K \begin{pmatrix} u^k \\ p^k \end{pmatrix} \right), \tag{12}$$

where

$$P = \begin{pmatrix} I & -D^{-1}B^T \\ 0 & I \end{pmatrix} \begin{pmatrix} F & 0 \\ B & S \end{pmatrix}^{-1} \tag{13}$$

and

$$K = \begin{pmatrix} F & B^T \\ B & 0 \end{pmatrix}. \tag{14}$$

We shall use one iteration of the SIMPLE method as a preconditioner for the GCR method applied to the discretized Navier-Stokes equations. More details about the SIMPLE preconditioner can be found in [22] and [24].

One of the many improvements of SIMPLE is called SIMPLER. In this method the first step is replaced by solving p^* from

$$Sp^* = r_p - BD^{-1}((D - F)u^k + r_v), \tag{15}$$

where u^k is obtained from the prior iteration. In case SIMPLER is used as preconditioner, u^k is taken equal to zero. One iteration of the SIMPLER algorithm is approximately 1.3 times more expensive than the SIMPLE iteration [24]. One step of the SIMPLER method is used as a preconditioner in combination with a Krylov subspace method.

To reduce the cost of the SIMPLER preconditioner, one might replace the matrix S in (15) by its diagonal. That method is denoted as SIMPLE-S. Table 1 gives an overview of these various SIMPLE implementations.

Step1: (SIMPLE)	Estimate pressure p^* (previous iterate)
Step1:(SIMPLER)	Solve $Sp^* = r_p - BD^{-1}((D - F)u + r_v)$, u from the previous iterate
Step1:(SIMPLE-S)	Solve $diag(S)p^* = r_p - BD^{-1}((D - F)u + r_v)$, u from the previous iterate
Step2:	Solve $Fu^* = r_v - B^T p^*$
Step3:	Solve $S\delta p = r_p - Bu^*$
Step4:	update $u = u^* - D^{-1}B^T\delta p$
Step5:	update $p = p^* + \delta p$

TABLE 1. Variants of SIMPLE method, one iteration is used as preconditioner

We also tried a variant of SIMPLE, which is in fact a combination of AL and SIMPLE. We call this method SIMPLE(AL). In this method we solve the augmented system (4) and replace F by $F = F + \gamma B^T W^{-1} B$. Besides that the approximation $\hat{S}^{-1} = -(\nu \hat{Q}_p^{-1} + \gamma W^{-1})$ is used. The advantage of this system is that no system of equations for the pressure has to be solved.

4.6. **ILU-type preconditioners.** A completely different approach, is to solve the coupled system (3) by a suitable iteration method. This is only possible if pivoting is applied, since the diagonal elements corresponding to the pressure unknowns are zero. The preconditioning used is standard ILU combined with a suitable ordering. Our method is to use a- priori reordering of unknowns, such that during computation of the ILU matrix, no zero pivots arise. The other method is to use pivoting during elimination. The first approach is possible by reordering the unknowns as follows: first number all the velocity unknowns, and then the pressure afterwards. However, a more clever reordering, based on the renumbering of nodal points, by

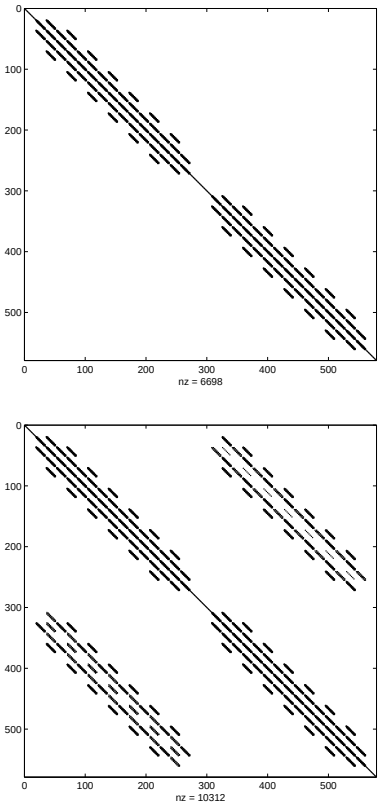

FIGURE 1. A1-Nonzero pattern of the coefficient matrix obtained from Picard iterations (Left), A2- Nonzero pattern of the coefficient matrix after adding $\gamma B^T W^{-1} B$ in the coefficient matrix (Right)

some standard method, and the definition of levels is possible. This method, SILU is described in [17].

The alternative is to use pivoting during elimination. To that end we applied the software package ILUPACK developed by Matthias Bollhöfer[1] [18]. This package creates an ILU preconditioner based on the following points:

- Static reordering of a matrix done with various efficient reordering schemes (like approximate minimum degree(AMD), reverse Cuthill McKee(RCM) etc).
- row and column scalings
- At each step of the ILU factorization, it is ensured that the inverse factors satisfy an upper bound $\mathcal{K}$, $\|\mathbf{L_k}^{-1}\| \leq \mathcal{K}$ and $\|\mathbf{U_k}^{-1}\| \leq \mathcal{K}$.

ILUPACK produces a robust and stable ILU factorization. A drop tolerance strategy is employed to reduce the amount of fill-in. As a consequence we can not predict the memory and fill-in a priori. However, in this package, the fill-in is limited to a certain extent. The approximate memory can be assigned. See [18].

[1]Department of mathematics, TU Berlin, http://www.math.tu-berlin.de/ilupack

5. Numerical Experiments

In this section we consider a number of numerical experiments. The Stokes and Navier-Stokes problems are solved in the following domains:

(1) The Poiseuille 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 $(-1, L) \times (-1, 1)$, known as the backward facing step shown in Figure 2. A Poiseuille 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.

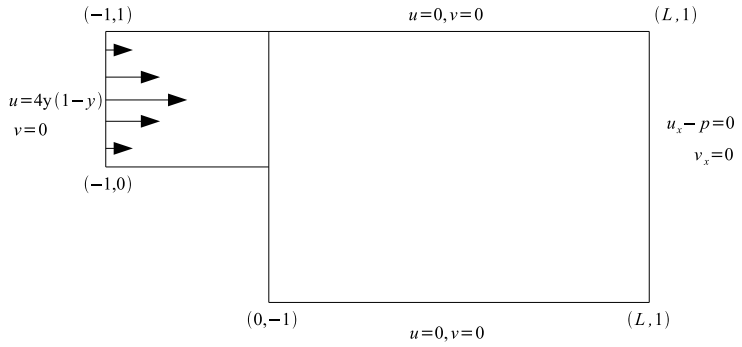

Figure 2. Backward facing step domain

Experiments are performed with preconditioned GCR, GMRES and Bi-CGSTAB methods. An ILU preconditioned GMRES or Bi-CGSTAB method is used for solving the linear system arising from the preconditioner, $Pz = r$. In the tables, "*in-it*" is defined as the number of inner iterations. This is the number of iterations to solve subsystems $Pz = r$. "*out-it*" is defined as the number of outer iterations. This is the number of iterations of the Krylov subspace method to solve $Ax = b$. In block preconditioners, we use a preconditioned method as inner solver. That means that we have a different preconditioner at each step of the Krylov subspace method. At each step of the outer Krylov subspace method, the inner iterative method is stopped after a certain number of iterations. This number can be either fixed or defined by some stopping criterium. Therefore some flexible type Krylov method like GMRESR or GCR must be used. We use the GCR method since it allows variable preconditioners and shows robust convergence behavior [22]. In the IFISS package, we can only test the preconditioner after a certain number of linearization steps. In our case, we do our experiments after the first step of the Picard linearization. The *Sloan reordering scheme* is used for the reordering of grid points. The grid size in the tables and figures refer to the number of nodes instead of the number of elements. The Q2-P1 elements are quadratic quadrilaterals with linear discontinuous pressure (Crouzeix Raviart elements). The Q2-Q1 Taylor-Hood elements have a "bilinear" continuous pressure approximation. The comparision of augmented type preconditioners and overall comparison is done on the system *Intel Dual Core, 2.66 GHz processor with 8GB RAM.* All other comparisons are made on *Intel Pentium 4, 2.66 GHz processor with 1GB RAM.*

First we compare the various SIMPLE type, augmented type and ILU type preconditioners separately and finally we do an overall comparison.

Comparison between SIMPLE and SIMPLER. In Table 2 the SIMPLE, SIMPLER, SIMPLE-S preconditioners are compared for the channel problem. Table 3 shows the same results for the backward-facing step. From Tables 2 and 3 it is clear that SIMPLER is the best choice of the three options. It can also be shown that SIMPLE type solvers perform better for Taylor-Hood elements than for Crouzeix Raviart elements. Hence, in the remainder we shall use the SIMPLER implementation as the only one of the SIMPLE family.

	SIMPLE	SIMPLER	SIMPLE-S
Grid	GCR outer iterations(time in seconds)		
Q2-P1			
9×9	12(0.05)	8(0.05)	10(0.04)
17×17	20(0.2)	11(0.16)	19(0.2)
33×33	34(1.8)	19(1.7)	32(1.64)
Q2-Q1			
9×9	11(0.03)	9(0.04)	11(0.04)
17×17	18(0.15)	12(0.14)	21(0.17)
33×33	28(0.94)	18(0.95)	38(1.13)

TABLE 2. Stokes channel flow problem, GMRES(1e-2(p),1e-1(u),1e-2(p)) for inner iterations, GCR(20) method with an accuracy = 10^{-4}

	SIMPLE	SIMPLER	SIMPLE-S
Grid	GCR outer iterations(time in seconds)		
Q2-P1			
9×25	54(0.39)	11(0.1)	22(0.17)
17×49	82(2.6)	11(0.46)	47(1.60)
33×97	137(29.93)	21(7.35)	111(27.63)
Q2-Q1			
9×25	32(0.2)	15(0.11)	20(0.17)
17×49	77(1.7)	17(0.41)	43(1.06)
33×97	119(19)	23(3.59)	108(17.24)

TABLE 3. Navier-Stokes backward facing step problem, Re=100, GMRES(1e-2(p),1e-1(u),1e-2(p)) for inner iterations, GCR(20) method with an accuracy = 10^{-4}

Comparison between AL and SIMPLE(AL). As mentioned earlier, a multigrid solver is suggested to solve the linear system $F_\gamma z_v = r_v - B^T z_p$ in the AL preconditioner. However, we are using a Krylov solver in combination with an ILU preconditioner. In some cases, the ILU preconditioner is used with Sloan reordering. Table 4 shows the results of the AL and SIMPLE(AL) preconditioner applied in combination with GCR. We have used the A1-nonzero pattern. It is clear that AL performs better than SIMPLE(AL). The same conclusion can be drawn from Table 5, where we solved the backward facing step problem with Reynolds number = 100. Note that with a reordering of grid points both methods become much faster due to a decrease of inner iterations.

	AL preconditioner			SIMPLE(AL) preconditioner		
Grid size	out-it	in-it	time(s)	out-it	in-it	time(s)
9×9	5	76	0.07	7	125	0.30
17×17	6	139	0.39	7	193	0.55
33×33	5	128	1.36	7	229	2.32
65×65	5	226	4.27	9	604	11.56

Table 4. Solution of the Navier-Stokes channel flow Problem, $Q_2 - P_1$ discretization with preconditioned GCR (*accuracy* of 10^{-4}), Picard linearization, inner iterations with GMRES(1e-2), A1 nonzero pattern, Re=100

	No reordering			with-reordering		
Grid size	out-it	in-it	time(s)	out-it	in-it	time(s)
AL preconditioner						
9×25	8	159	0.28	8	149	0.260
17×49	10	164	1.40	10	127	0.92
33×97	7	541	6.45	8	312	3.97
SIMPLE(AL) preconditioner						
9×25	9	186	0.35	9	183	0.35
17×49	9	329	2.24	9	249	1.75
33×97	9	734	8.93	9	380	5.05

Table 5. Solution of the Navier-Stokes backward facing step problem, $Q_2 - P_1$ discretization with preconditioned GCR (*accuracy* of 10^{-4}), Picard linearization, inner iterations with GMRES(1e-2),A1 nonzero pattern, Re=100

Comparison between the AL, AC and GD preconditioners. In Table 6, we have solved the Navier-Stokes problem with the GD and AC preconditioners for the Q2-P1 discretization. All preconditioners give nice convergence for a problem solved on a stretched grid. However, unlike the AL preconditioner, the AC and GD preconditioners are more sensitive to the choice of the parameter γ. Even with $\gamma = 16$, the number of iterations with AC and GD preconditioners increases with the increase in number of grid elements. Increasing the value of γ make it more difficult to solve the subsystem $F_\gamma z_v = r_v$. The AL preconditioner shows a robust behavior with $\gamma = 1$ for which AC and GD preconditioner fail to give desired convergence. With $\gamma = 16$, the AC preconditioner is competing with the AL preconditioner. Therefore, AL preconditioner is preferred due to its less dependence on the parameter γ. The memory requirements for these preconditioners are almost equal to that of the AL preconditioner, but the AL preconditioner makes use of an augmented system.

A comparison is also made between these preconditioners for the Q2-Q1 discretization. The AL preconditioner gives faster convergence than the AC and GD preconditioners.

	$\gamma = 1$			$\gamma = 16$		$\gamma = 256$	
Grid	AL	AC	GD	AC	GD	AC	GD
	GCR(20) iterations-time in seconds						
9×25	7(0.13)	16(0.98)	23(0.91)	5(0.34)	8(0.4)	4(0.5)	6(0.8)
17×49	7(0.30)	35(2.96)	NC	8(0.91)	12(0.97)	4(0.92)	7(2.13)
33×97	7(3.4)	91(26)	NC	14(4.36)	23(6.39)	5(4.0)	8(7.26)
65×193	8(59)	NC	NC	NC	NC	8(54)	12(83)

TABLE 6. Comparison of the AL, AC and GD preconditioners in the backward facing step Q2-P1 grid for the Navier-Stokes problem with Re=100, GCR for outer iterations with accuracy=10^{-4} and GMRES for inner iterations with an accuracy = 10^{-2}

Some remarks about the ILU preconditioner used in AL. ILU is a cheap and a simple preconditioner to implement. Moreover, the preconditioner gives a good performance for medium range problems. Some reordering and pivoting techniques make it more powerful and increases the band of convergence. However, in case of the augmented Lagrangian preconditioner, we have seen that solving the problem $F_\gamma z_v = r_v$ with an increasing value of γ, an ILU preconditioner does not give desirable convergence. This is true even for small problems. Though with relatively small γ the AL preconditioner converges, it takes a large number of outer iterations. In our experiments γ is taken to one. We have seen that lumping in $Q2-Q1$ and using the actual matrix pattern in $Q2-P1$ works fine in some cases. Here, we want to investigate the causes of failure of an ILU preconditioner for the problems that are solved with the augmented Lagrangian preconditioner with a moderate value $\gamma = 1$.

We know that the ILU preconditioner performs better if the system matrix is diagonally dominant. The addition of the term $\gamma B^T W^{-1} B$ to F introduces a large number of off diagonal nonzero entries. If F is SPD and diagonally dominant, F_γ with $\gamma > 0$ will still be SPD but this makes the system less diagonally dominant because some of the entries that are introduced are positive [30]. ILU in this case will strongly depend on the factor γ and the entries dropped. According to [14], there are three major causes when ILU breaks down:

(1) Inaccuracy: With inaccuracy we mean how far is the incomplete LU matrix from the system matrix

$$F = LU + R,$$

where R consists of the dropped entries. There are two major reasons why R can be large. One is due to inaccuracies caused by the dropped entries in the ILU factors. The other one is due to small pivots which cause elements to grow considerably due to the long recurrence associated with solving ILU factors. In the first case adding extra fill-in may imrpove convergence. In the second case this will not help at all.

(2) Zero pivot: In some cases we face a problem due to the occurence of a zero pivot. The problem can be solved by pivoting or a suitable reordering method.

(3) Unstable ILU factors: This problem arises if the matrix is far from diagonally dominant. The stability of the ILU factors can be checked by $\|(LU)^{-1}e\|_\infty$ where e is a vector consisting of ones. We will refer to this quantity as condest. This is also known as a condition estimate of $(LU)^{-1}$. If there are small pivots, the condition estimate will be large.

If $\frac{1}{\text{smallest pivot}}$ is comparable to condest, and both quantities are large, then it is assumed that at least one pivot is small. Otherwise if condest is much larger than $\frac{1}{\text{smallest pivot}}$, we assume long recurrences are the cause of the large condest.

Another comparison is made with the largest entry in the L and U factors. if $max(L+U)$ is much larger than $max(F)$, this also gives an indication of unstable factors. Usually the order of $\frac{1}{\text{pivot}}$ and $max(L+U)$ is the same. However, if $\frac{1}{\text{pivot}}$ is large and $max(L+U)$ is small, that means that a small pivot is not used in factorization [14].

All these problems can occur together. A factorization that is initially inaccurate due to dropping can produce small pivots that in turn make the factorization unstable and more inaccurate. For more details see [14].

In Table 7 we report the above described quantities for a number of problems. We see that even with the A2 nonzero pattern that has a higher number of non-zero elements than the A1 pattern, we do not get convergence in some cases. The reason with the A2 nonzero pattern is - due to long recurrence - that small pivots play a role

to make the incomplete LU factors unstable. Hence, inaccuracy ($\|R\|_\infty$) increases. In some cases lumping can improve the convergence. We recommend lumping should only be used when there is no convergence. Note that $max(L+U)$ and condest are reliable indicators of convergence. If $max(L+U)$ is much larger than $max(F_\gamma)$ and condest is large, the preconditioner is not useful because no convergence occurs. In such a case it is advised to use lumping in the preconditioner.

Grid(nonzero	$max(1/pivot)$	$max(L+U, F_\gamma)$	condest	$\|R\|_\infty$	Remarks
pattern)	Q2-P1				
9×25(A1)	2.55	3.48, 2.48	2 5	3.2	8(0.12)
(A2)	177	270, 2.48	10^5	225	NC(Unstable solve)
17×25(A1)	3.08	5.88, 4.88	27	3.4	8(0.28)
(A2)	203	10^4, 4.88	10^8	10^3	NC(Unstable solve)
lump(A2)	0.66	8.61, 4.88	4.44	8.87	7(0.71)
33×25(A1)	3.31	10.7, 9.7	49	3.5	8(0.63)
(A2)	10^3	10^3, 9.7	10^{15}	10^4	NC(Unstable solve)
lump(A2)	1.01	15.84, 9.7	5.3	14.7	8(2.54)
	Q2-Q1				
9×25(A1)	7.23	7.23, 1	27.3	1.87	9(0.27)
(A2)	1.15	2, 1	53	0.61	9(0.1)
17×25(A1)	32	32, 1.03	10^5	4.8	NC(inaccuracy)
(A2)	1.65	2, 1.03	261	0.97	9(0.44)
33×25(A1)	10^5	10^5, 2.01	10^5	10^4	NC(inacurracy)
(A2)	700	10^3, 2.01	10^7	10^3	NC(Unstable solve)
lump(A2)	0.67	5.60, 2.01	17	6.2	9(2.69)
lump(A) means nonzero pattern of A after lumping NC= no convergence					

TABLE 7. Various parameters comparison with AL preconditioner in a backward facing step

In order to compare ILUPACK and SILU, we have solved the Stokes problem with a Q2-Q1 and Q2-P1 discretization. In ILUPACK we used the following options

```
options.matching=1;
options.ordering='amd';
options.x0=zeros(n,1);
options.droptol=1e-2;
options.condest=1e2;
options.restol=1e-6;
options.maxit=500;
options.elbow=20;
options.lfil=n+1;
options.lfilS=n+1;
GMRES with nRestart =30
```

The following definitions are used:
$\mathcal{S}(0)$: SILU without fill-in,
$\mathcal{S}(1)$: SILU with extra fill-in,
$\mathcal{IP}(0)$: ILUPACK with the same fill-in as in $\mathcal{S}(0)$,
$\mathcal{IP}(1)$: ILUPACK with the same fill-in as in $\mathcal{S}(1)$ and
MR (memory ratio)/growth factor: This defines the ratio between the size of the ILU matrix computed by ILUPACK/SILU and the size of the coefficient matrix.

Figure 3 compares the results of ILUPACK and SILU. It is clear that ILUPACK requires less iterations than SILU. However, the CPU time of SILU is always less. The reason being the fill-in growth factor of ILUPACK which is shown in Figure 4. In Table 8, growth factor is brought to the value that is used in SILU. Again we see that the results of both preconditioners are comparable. Moreover, SILU performs better with the Bi-CGSTAB method, however, due to unavailability of the solver in ILUPACK, we are experimenting only with the GMRES method. Most of the CPU time in ILUPACK is spent in generating ILU factors. It shows robustness in generating stable ILU factors. The increase in the amount of work and memory is very large with the increase in problem size. The reason for this is that ILUPACK involves pivoting, scaling and extra fill-in in the ILU factors. Thus it consumes large CPU time in computing ILU factors compared to time consumed in computing the solution.

The results given in this section is done with Q2-Q1 and Q2-P1 rectangular elements. However, we have tested these preconditioners for triangular elements as well and observed similar results as that in rectangular elements.

Grid	$\mathcal{S}(0)$		$\mathcal{IP}(0)$		MR	$\mathcal{S}(1)$		$\mathcal{IP}(1)$		MR
	Iter.	Time(s)	Iter.	Time(s)	MR	Iter.	Time(s)	Iter.	Time(s)	MR
Q2-Q1 rectangular elements										
33×97	56	0.27	34	0.54	1.73	19	0.23	17	0.60	2.78
65×193	207	3.1	83	3.70	1.79	54	1.76	34	3.15	2.80
Q2-P1 rectangular elements										
33×97	143	0.55	35	0.67	1.77	22	0.3	19	0.73	2.98
65×193	482	7.2	75	3.86	1.73	56	1.96	29	4.1	2.95

Table 8. Comparison with matched memory ratio

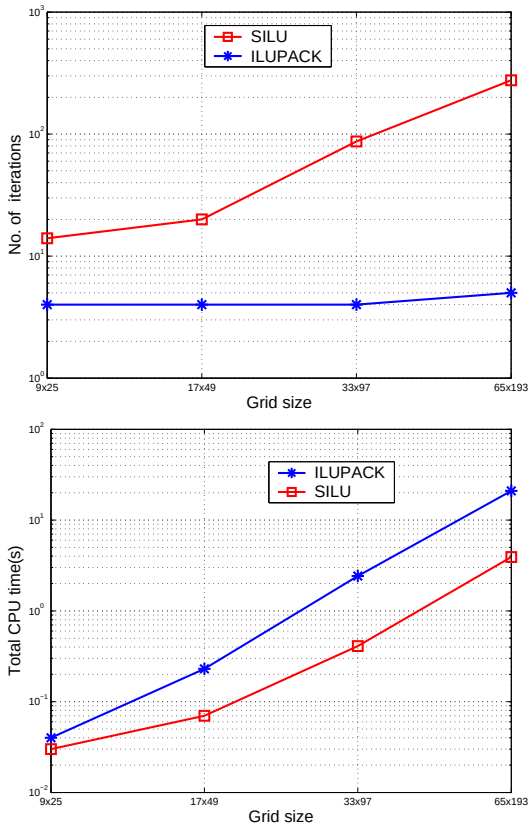

FIGURE 3. Solution of the Stokes backward facing step problem with the preconditioned GMRES(20), number of iterations for the ILUPACK and SILU preconditioner (Left), CPU time in seconds (Right)

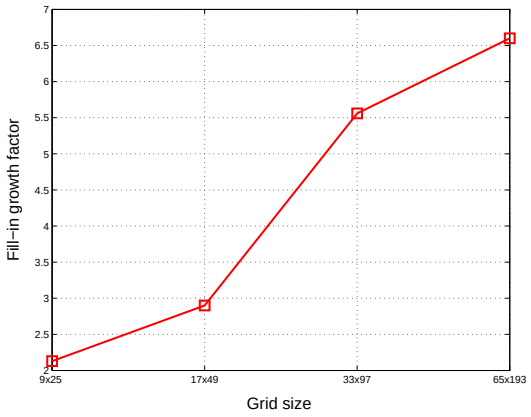

FIGURE 4. Fill-ins growth factor in ILUPACK

SILU convergence. From the results in [17] and the previous sections, we conclude that the SILU preconditioner costs less CPU time compared to the other preconditioners for a certain range of problems in 2D. The convergence of SILU can sometimes be increased by allowing fill-in [17]. In general, SILU detoriates with the increase in problem size and Reynolds number. Therefore we introduce fill-in in the pressure part of the continuity equation. The fill-in pattern is shown in Figure 5. Results given in Table 9, show that for small problems, fill-in in the pressure part works fine. But the method is not recommended since the convergence becomes worse in some cases.

We have analyzed the effect of stretching on the eigenvalues of the SILU preconditioner. In Table 10, we see that an increase in length broadens the eigenvalues spectrum of the coefficient matrix A. Some of the eigenvalues becomes closer to zero and some becomes more negative. However the number of eigenvalues with negative real part and positive real part remain the same. In the preconditioned system $P^{-1}A$ stretching has a bad effect on the eigenvalues spectrum. The eigenvalue spectrum for $P^{-1}A$ for various lengths of the channel is shown in Figure 6. From this figure it appears that some eigenvalues are close to zero and even shifted from positive to negative in some cases. This in turn deteriorates the convergence of Krylov subspace methods.

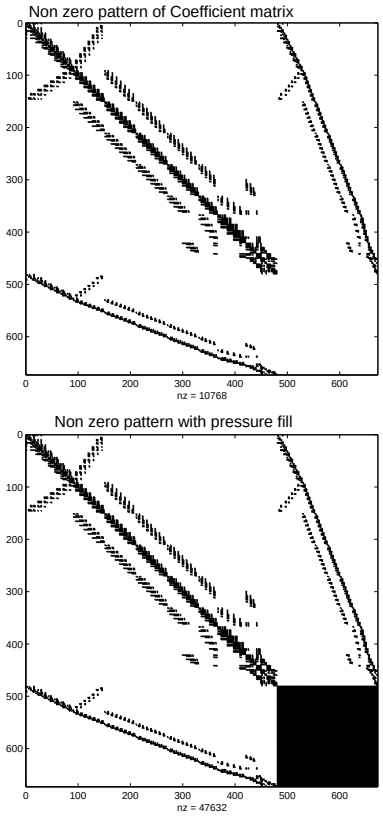

FIGURE 5. Nonzero pattern of the coefficient matrix (Left), Nonzero pattern after fill-ins at pressure points in the continuity equation (Right)

grid	Pressure-fill	$nnz(A)$	$nnz(L+U)$	iter
Picard				
9×25	No	3720	7440	39.0
9×25	Yes	4440	8880	20.0
17×25	No	8544	17088	55.0
17×25	Yes	10648	21296	19.0
33×25	No	18364	36728	99.0
33×25	Yes	27890	55780	NC
33×25	No	18364	36728	297.0 GMRES
33×25	Yes	27890	55780	120.0 GMRES
65×25	No	38380	76760	372.0 GMRES
65×25	Yes	46382	92764	600.0 GMRES

TABLE 9. Navier-Stokes backward facing step, Q2-P1, Re=100, Bi-CGSTAB(10^{-4})

length	min λ	max λ	min $\|\lambda\|$
L=5	-0.19	18.5	0.004
L=10	-0.34	33.7	0.0023
L=20	-0.63	64.3	0.0012
L=40	-1.2	125	0.0006
L=100	-3.1	309	0.0003

TABLE 10. Eigenvalues of the coefficient matrix with increasing length

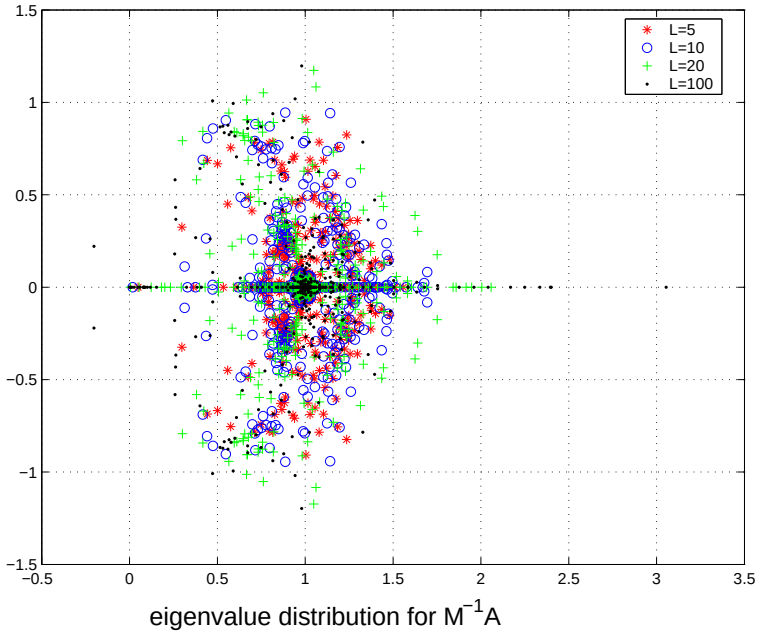

FIGURE 6. Eigenvalues of $P^{-1}A$ for SILU preconditioner with increasing length

Overall comparison. In this section, we compare the preconditioners for the Navier-Stokes problem that we treated in the previous section.

Table 11 shows results with a $Q2-Q1$ and $Q2-P1$ discretization. In iteration count, we see that AL and SIMPLE(AL) show faster convergence than the other preconditioners mentioned in the table. SIMPLE(AL) seems to be two times more expensive than AL in CPU time. In $Q2-Q1$, these two preconditioners are expensive due to the addition of an extra term in the velocity matrix. However in $Q2-P1$, AL results shows a very good compromise between the number of iterations and the CPU time. The good thing about the AL preconditioner is its mesh and Reynolds number independent convergence.

In terms of the CPU time, SILU performs better than the other preconditioners mentioned in Table 11. However the number of iterations with the SILU preconditioner increases with the increase in number of grid elements. The number of iterations can be reduced by using extra fill-in but it makes the preconditioner more expensive because it consumes large amount of memory and CPU time.

The convergence of SIMPLER and LSC can be categorized almost between AL and SILU. Both SIMPLER and LSC consume more iterations than AL and less than SILU. In terms of CPU time, it is the other way around. The outer iterations of block preconditioners may be constant, but the number of inner iterations increases with the increase in grid size if an ILU preconditioner is used.

The block preconditioners that are discussed in this section have the common property that their convergence is independent of the mesh size and the Reynolds number. There are two reasons for the increase in the CPU time in these block preconditioners. One reason is the increase in problem size, and the other is due to increase in the number of inner iterations. Therefore to use the block preconditioners efficiently, a demand for better inner solvers always exists. A multigrid solver can reduce the CPU time of this block preconditioner. This is certainly the case if one uses the AL preconditioner [27].

For the AL and SIMPLE(AL) preconditioners, with $Q2-Q1$ discretization convergence in some cases is only possible if one uses the A2- non-zero pattern with lumping. In lumping, all the off diagonal elements that have the same sign as that of diagonal element are added to the diagonal and the elements themselves are made zero. The ILU decomposition of the lumped matrix is made by using the nonzero pattern of the lumped matrix.

Stretching. In Table 12, we compare the preconditioners using stretched grids. A channel flow domain with a variable length in the flow direction is used to solve the Navier-Stokes problem with Re=200. Results show that none of the preconditioner give convergence independent of stretching. However, some of these preconditioners show very good behavior with grid stretching.

With Q2-Q1 discretization, LSC seems to be a better choice to use in stretched grids. The SIMPLER preconditioner also performs better than the AL and SIMPLE(AL) preconditioners. However, if a comparison is made between LSC and SIMPLER, than LSC is the better option due to the fact that its convergence is less dependent on the grid size. Moreover, we see that LSC convergence is faster than SIMPLER. The time comparison is not realistic in cases where * is used. * is used where the MATLAB built-in ILU preconditioner based on drop tolerance is used. For the other cases we use ILU without fill-in. SILU performs poor and does

Grid	AL	SIMPLE(AL)	SIMPLER	LSC	SILU
GCR(20) iterations-time in seconds					
Q2-Q1					
9×25	9(0.16)	10(0.34)	14(0.22)	15(0.19)	17(0.02)
17×49	8(2.55)	10(4.9)	15(0.74)	13(0.65)	19(0.05)
33×97	8(23.17)	8(45)	17(6.92)	11(4.22)	36(0.37)
65×193	8(128)	8(265)	33(94)	16(46)	154(6.0)
Q2-P1					
9×25	7(0.13)	8(0.11)	11(0.15)	14(0.23)	61(0.1)
17×49	7(0.30)	8(0.54)	11(0.39)	12(0.5)	79(0.38)
33×97	7(3.34)	8(7.39)	15(5.68)	11(5.18)	100(1.79)
65×193	8(59)	8(108)	24(28)	16(19)	205(8.0)

TABLE 11. Comparison of various preconditioners in the backward facing step for the Navier-Stokes problem with Re=100, and GCR method, lumping used in the AL and SIMPLE(AL) preconditioners in the Q2-Q1 discretization

not show convergence for highly stretched grids.

In Q2-P1, we see that AL and SIMPLE(AL) appear to be much better options than the other preconditioners. However, the number of outer/inner iterations increases with the increase in stretching. Other preconditioners fail to perform better for highly stretched grids.

With a direct solver used as inner solver in these block preconditioner, we can get convergence in all preconditioners. However, using direct solver is not of practical use in large problems.

Grid	AL	SIMPLE(AL)	SIMPLER	LSC	SILU
GCR(20) iterations-time in seconds					
33×33 Q2-Q1					
$L = 2$	4(1.85)	6(3.4)	19(0.73)	19(0.6)	22(0.09)
$L = 10$	7(5.13)	8(8)	46(1.69)	34(1.14)	57(0.26)
$L = 50$	23(106)	24(133)	74(9.59)	41(1.40)	NC
$L = 100$	36(16)*	36(18)*	71(9.4)	42(1.44)	NC
$L = 200$	47(28)*	49(34)*	73(9.7)	42(1.44)	NC
33×33 Q2-P1					
$L = 2$	5(0.88)	6(1.17)	12(1.04)	16(1.17)	117(1.16)
$L = 10$	10(1.38)	10(1.29)	15(0.91)	21(1.72)	340(4.96)
$L = 50$	29(1.92)	29(1.92)	36(3.76)*	39(4.63)*	NC
$L = 100$	29(1.01)	30(1.70)	NC	77(25)*	NC
$L = 200$	50(2.05)	51(2.35)	NC	NC	NC

TABLE 12. Comparison of various preconditioners in the channel domain stretched in flow direction for the Navier-Stokes problem with Re=200, and GCR method

Memory requirements for the preconditioners. The Navier-Stokes problem given in matrix notation after linearization is:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}, \tag{16}$$

Since all the preconditioners except SILU use the submatrix F and the matrix B or B^T, we exclude this memory requirement in our memory calculations. We also exclude the memory requirements to store z and r. The subproblems are solved with a direct solver or an ILU preconditioner. Therefore, we include memory requirements of the ILU matrices. Below we give some definitions.

Definition 2: For calculating memory, we define ILU(F) as an incomplete LU decomposition of matrix F, ILU(S) as an incomplete LU decomposition of the Schur complement operator and ILU(sys) as that of system matrix. n is the number of velocity unknowns and m is the number of pressure unknowns.
b_F is the average nonzero elements per row of F. b_S, b_{sys} and b_{aug} have the same definition for the Schur complement, system matrix and augmented matrix, respectively.

Table 13 shows the general memory requirements of a preconditioner, if the subsystems are solved with an ILU preconditioner. In Table 14 and 15, we calculate the memory required for each preconditioner for the 9×25 and 33×97 grids discretized by Q2-Q1 and Q2-P1 finite elements.

On the basis of these calculations, we see that memory requirements for each preconditioner are strongly dependent on the type of discretization and linearization scheme that we use. From a memory point of view, SILU requires lesser memory than the other preconditioners. It is not a good idea to use AL or SIMPLE(AL) for problems discretized by Q2-Q1 elements because the matrix, which arises due to the coupling between velocities, introduces a large number of nonzero elements in the matrix F. Compared to the Picard method, the Newton linearization - due to the addition of the Newton derivative matrix - gives rise to approximately two times the memory requirement in SIMPLER, SILU and the LSC preconditioner. However, in AL type preconditioners -due to the addition of an extra term in F - the memory requirement with both linearization schemes is the same. In case of a non-constant viscosity the space needed for Newton and Picard is always the same, since we can not use the reduced stress tensor. Since the memory requirement of AC and GD preconditioner are almost the same as that of the AL preconditioner, they are not included in the tables.

Type of matrix	AL	SIMPLE(AL)	SIMPLER	LSC	SILU
ILU(F)	$b_{aug}.n$	$b_{aug}.n$	$b_F.n$	$b_F.n$	-
ILU(S)	-	-	$b_S.m$	$b_S.m$	-
S	m	m	$b_S.m$	$b_S.m$	-
vector	-	-	m	n	-
ILU(sys)	-	-	-	-	$b_{sys}.(m+n)$

TABLE 13. Memory requirements of various preconditioners

Type of matrix	AL	SIMPLE(AL)	SIMPLER	LSC	SILU
ILU(F)	53×418	53×418	9×418	9×418	-
ILU(S)	-	-	17×61	17×61	-
S	61	61	17×61	17×61	-
vector	-	-	61	418	-
ILU(sys)	-	-	-	-	15×479
9×25(Picard)	22215	22215	5897	6254	7185
other examples					
33×97 (Picard)	2535369	2535369	487905	508002	345562
33×97 (Newton)	2541069	2541069	832693	852666	690226

TABLE 14. Memory requirements in double precision numbers for the preconditioners used in solving the Navier-Stokes backward facing step problem for different size Q2-Q1 grid, Re=100

Type of matrix	AL	SIMPLE(AL)	SIMPLER	LSC	SILU
ILU(F)	16×418	16×418	9×418	9×418	-
ILU(S)	-	-	20×132	19×132	-
S	132	132	20×132	19×132	-
vector	-	-	132	418	-
ILU(sys)	-	-	-	-	12×550
9×25(Picard)	6820	6820	9174	9196	7185
other examples					
33×97 (Picard)	688600	688600	795144	806460	345562
33×97(Newton)	698674	698674	1141114	1151124	690226

TABLE 15. Memory requirements in double precision numbers for the preconditioners used in solving the Navier-Stokes backward facing step problem for different size Q2-P1 grid, Re=100

6. Conclusions

The Navier Stokes problem discretized by a finite element method is solved with a preconditioned Krylov subspace method. Results are given with various preconditioners which are popular for the incompressible Navier-Stokes problem. We combine the SIMPLE method with the augmented Lagrangian matrix: SIMPLE(AL). The approximation used in the augmented Lagrangian preconditioner is used in the SIMPLE preconditioner. With this approximation, convergence of the SIMPLE preconditioner improves and becomes comparable with AL.

Results for SILU and ILUPACK are compared. At the cost of large construction time required to build ILU factors, ILUPACK gives faster convergence than SILU if more fill-in is allowed in ILUPACK. We can assign predefined memory in SILU, however since ILUPACK is based on a drop tolerance strategy it is hard to determine memory costs beforehand . In fact a priori memory is computed in ILUPACK by allowing certain amount of maximum fill-in per row. By using same amount of fill-in in ILUPACK and SILU, SILU performs better than ILUPACK.

LSC is a better preconditioner than SIMPLER. Both involve two Poisson solves with different approximations. However, as the problem size increases, the matrix vector product with the full matrix in the LSC preconditioner becomes more expensive. Compared to the AL preconditioner, the AC and GD preconditioner strongly depend on the values of γ. AL preconditioner with $\gamma = 1$ converges faster than the AC and GD preconditioner.

In the overall comparison, SILU performs better than the other preconditioners. However, the performance of SILU becomes poor when used in stretched grids. For stretched grid, AL and LSC are better options to use in Q2-P1 and Q2-Q1 discretizations, respectively.

References

[1] F. Brezzi, M. Fortin. *Mixed and hybrid finite element methods*. Springer-Verlag, New York, 1991.

[2] C. Dohrmann, P. Bochev. A stabilized finite element method for the Stokes problem based on polynomial pressure projections. *Int. J. for Num. Meth. in Fluids, 46*. 183-201, 2004.

[3] M. Fortin. Old and new finite elements for incompressible flows. *Int. J. for Num. Meth. in Fluids, 1*. 347-364, 1981.

[4] J. A. Meijerink, H. A. van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comput. 31*. 148, 1977.

[5] C. Taylor, P. Hood. A numerical solution of the Navier Stokes equations using the finite element technique. *Comput. Fluids, 1*. 73-100, 1973.

[6] M. Crouzeix , P. A. Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. *RAIRO, Ser. Rouge Anal. Num. 3*, 33-76, 1973.

[7] K. Arrow, L. Hurwicz. *Studies in non-linear programming*. Stanford university press, Stanford, CA, 1958.

[8] C. Cuvelier, A. Segal, A.A. van Steenhoven. *Finite Element Methods and Navier Stokes Equations*. Reidel Publishing Company, Dordrecht, Holland, 1986.

[9] Y. Saad, M. H. Schultz. GMRES: A generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comput. 7*, 856-869, 1986

[10] H. A. Van Der Vorst. Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems. *SIAM J. Sci. Stat. Comput. 13*, 631-644, 1992.

[11] H. A. Van Der Vorst, C. Vuik. The superlinear convergence behavior of GMRES. *J. Comput. Appl. Math. 48*, 327-341, 1993.

[12] H. C. Elman, D. Silvester. Fast nonsymmetric iterations and preconditioning for Navier Stokes equations. *SIAM J. Sci. Comput. 17*, 33-46, 1996.

[13] S. W. Sloan. An algorithm for profile and wave front reduction of sparse matrices. *Int. J. for Num. Meth. in Engng. 23*, 239-251, 1986.

[14] E. Chow and Y. Saad. Experimental study of ILU preconditioners for indefinite matrices. *J. Comput. Appl. Math. 86* , 387-414 ,1997.

[15] H. C. Elman, V. E. Howle, J. Shadid, R. Shuttleworth, R. Tuminaro. Block preconditioner based on approximate Commutators. *SIAM J. Sci. Comput., 27*. 1651-1667, 2006.

[16] S. C. Eisenstat, H. C. Elman, M. H. Schultz. Variational Iterative Methods for Nonsymmetric Systems of Linear Equations.
SIAM Journal on Numerical Analysis, Vol. 20, No. 2, 345-357, 1983.
[17] M. ur Rehman, C. Vuik, G. Segal. A comparison of preconditioners for incompressible Navier-Stokes solvers. To appear in *Int. J. for Num. Meth. in fluids*, 2007.
[18] M. Bollhöfer, Y. Saad. Multilevel preconditioners constructed from inverse-based ILUs. *SIAM J. Sci. Comput. 27*, 1627-1650, 2006.
[19] M. Benzi, G. H. Golub, J. Liesen. Numerical solution of saddle point problems. *Acta Numerica*, 1-137, 2005.
[20] SV. Patankar. *Numerical heat transfer and fluid flow*. McGraw-Hill, New York, 1980.
[21] P. Wesseling. *Principles of Computational fluid dynamics* . Springer Series in Computational Mathematics, vol.29. Springer, Heidelberg, 2001.
[22] C. Vuik, A. Saghir, GP. Boerstoel. The Krylov accelerated SIMPLE(R) method for flow problems in industrial furnaces. *Int. J. Numer. Methods in fluids. 33*, 1027-1040, 2000.
[23] C. Vuik, A. Saghir . The Krylov accelerated SIMPLE(R) method for incompressible flow. *Technical Report TUDelft*, 02-01, 2002.
[24] C. Li, C. Vuik. Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow. *Numer. Linear Algebra Appl. 11*, 511-523, 2004.
[25] M. Benzi. Preconditioning techniques for large linear systems. *J. Comp. Physics, 182*, 418-477, 2002.
[26] H. C. Elman. Preconditioning for the steady-state Navier Stokes equations with low viscosity. *SIAM J. Sci. Comput. 20*, 1299-1316, 1999.
[27] M. Benzi, M. A. Olshanskii, An Augmented Lagrangian-Based Approach to the Oseen Problem. *SIAM J. Sci. Comput. 28*, 2095-2113, 2006.
[28] M. F. Murphy, G. H. Golub and A. J. Wathen. A note on preconditioning for indefinite linear systems, *SIAM J. Sci. Comput. 21*, 1969-1972, 2000.
[29] H. C. Elman, D. Silvester, A. J. Wathen. *Finite Elements and Fast iterative solvers with applications in incompressible fluids dynamics*. Oxford University Press, Oxford, 2005.
[30] A. C. Niet and F. W. Wubs. Two preconditioners for saddle point problems in fluids flow, *Int. J. Numer. Methods in fluids. 54(4)* 355-377, 2007