DELFT UNIVERSITY OF TECHNOLOGY

REPORT 06-05

Solution of the incompressible Navier Stokes equations with preconditioned Krylov subspace methods

M. ur Rehman, C. Vuik G. Segal

ISSN 1389-6520

Reports of the Department of Applied Mathematical Analysis

Delft 2006

SOLUTION OF THE INCOMPRESSIBLE NAVIER STOKES EQUATIONS WITH PRECONDITIONED KRYLOV SUBSPACE METHODS

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

Abstract. In this report, using SEPRAN [1], the incompressible Stokes and Navier Stokes problems are solved in a square and L-shaped domain with a varying grid size and Reynolds number. Both problems are solved with iterative solvers using an ILU preconditioner. Different ordering techniques of the grid points and the unknowns are used to avoid breakdown of the LU decomposition. These ordering techniques used with ILU preconditioning makes that the iterative methods applied to the system of equations converge rapidly. With the reordering techniques, a direct solver can be used to solve the coupled system without pivoting. Numerical experiments are performed in a 2-D and 3-D domain using various finite element discretization schemes.

Contents

List of Figures

List of Tables

1. Introduction

In this report, the incompressible Navier Stokes equations are first discretized by a finite element method. One obtains a nonlinear system having two or three velocity unknowns and pressure. The nonlinear system arising from the discretization is first linearized, which gives rise to a nonsymmetric indefinite linear system of equations. The linear system is then solved with a direct or iterative solver both in 2-D and 3-D.
In Section 2, we discretize the Navier Stokes equations by a finite element method. The techniques used in linearization of Navier Stokes equations are discussed. The number of rows of the continuity equation is determined by the the number of pressure unknowns, more velocity unknowns than pressure unknowns are demanded to avoid dependent or inconsistent systems of equations. The choice of the element determines whether this requirement is satisfied. This issue is also discussed in this section.
In Section 3, we discuss solution strategies for a linear system like direct, classical iterative method and Krylov subspace methods. We give a general overview of these methods while some of them which are specific to our problem are discussed.
In section 4, we describe a preconditioning strategy, which is based on a renumbering of the finite element grid and reordering of the velocity and pressure unknowns. This scheme is helpful for both direct and iterative methods. In a direct method, it avoids the break down in the LU decomposition due to zeros at the main diagonal and reduces fill-in. When using an ILU preconditioner with these reordering schemes, the convergence of Krylov subspace method is in general enhanced.
In Section 5, based on the given scheme, we perform some numerical experiments both in 2-D and 3-D. For a some small problems direct methods are more efficient than iterative methods, but for medium and large problems a preconditioned Krylov subspace method gives faster results in CPU time than a direct method.

2. Incompressible Navier Stokes Equations

In this chapter we are concerned with the basic equations of fluid dynamics and its discretization. We start with the steady state Navier Stokes equations governing the flow of a Newtonian, incompressible viscous fluid. The equations are given by

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

$$\nabla.\vec{u} = 0\ in\ \Omega. \tag{2}$$

Where $\Omega \subset \mathbf{R}^d(d = 2\ or\ 3)$ is the flow domain with piecewise smooth boundary $\partial\Omega$, $\vec{u}$ is the fluid velocity, p is the pressure field, $\nu > 0$ is the kinematic viscosity coefficient (inversely proportional to Reynolds number Re), Δ is the Laplace operator, ∇ denotes the gradient and $\nabla.$ is the divergence.
Equation (1) represents conservation of momentum, while equation(2) represents the incompressibility condition, or mass conservation. The boundary value problem that is considered is the system (1,2) posed on a two or three dimensional domain Ω, together with boundary conditions on $\partial\Omega = \partial\Omega_D \cup \partial\Omega_N$ given by

$$\vec{u} = \vec{w}\ on\ \partial\Omega_D,\ \nu\frac{\partial\vec{u}}{\partial n} - \vec{n}p = \vec{s}\ on\ \partial\Omega_N.$$

Here we will restrict ourselves to the simplest case, that is no external force: $\vec{f} = 0$ and fixed wall boundary conditions, $\vec{u} = 0\ on\ \partial\Omega$.
The presence of the convective term $\vec{u}.\nabla\vec{u}$ in the momentum equation makes the Navier Stokes system non-linear. It can be linearized with Picard or Newton's method. We will discuss this later. In the limiting case when the convection is

negligible ($\nu \to \infty$), the Navier Stokes equations reduce to the Stokes equations given by

$$-\nabla^2\vec{u} + \nabla p = \vec{f}\ in\ \Omega \tag{3}$$

$$\nabla.\vec{u} = 0\ in\ \Omega, \tag{4}$$

with boundary condition

$$\vec{u} = \vec{w}\ on\ \partial\Omega_D,\ \frac{\partial\vec{u}}{\partial n} - \vec{n}p = \vec{s}\ on\ \partial\Omega_N.$$

2.1. **Discretization.** The discretization of the Navier Stokes equations is done through the finite element method. The weak formation of the Navier Stokes equations ia given as:

$$\nu\int_\Omega(\nabla^2\vec{u}).\vec{v} + \int_\Omega(\vec{u}.\nabla\vec{u}).\vec{v} - \int_\Omega(\nabla p).\vec{v} = 0, \tag{5}$$

$$\int_\Omega q(\nabla.\vec{u}) = 0, \tag{6}$$

where $\vec{v}$ and q are the test functions. Applying the Gauss divergence theorem and substituting the boundary conditions in (5) and (6), it reduces to

$$\nu\int_\Omega\nabla\vec{u} : \nabla\vec{v} + \int_\Omega(\vec{u}.\nabla\vec{u}).\vec{v} - \int_\Omega p(\nabla.\vec{v}) = \int_{\partial\Omega_N}\vec{s}.\vec{v}, \tag{7}$$

$$\int_\Omega q(\nabla.\vec{u}) = 0. \tag{8}$$

A discrete weak formulation is defined using finite dimensional subspaces $\mathbf{X}_0^h \subset \mathbf{H}_{E_0}^1$ and $M^h \subset L_2(\Omega)$, given a velocity solution space $\mathbf{X}_E^h$, the discrete version of (7) and (8) is: find $u_h \in \mathbf{X}_E^h$ and $p_h \in M^h$ such that

$$\nu\int_\Omega\nabla\vec{u_h} : \nabla\vec{v_h} + \int_\Omega(\vec{u_h}.\nabla\vec{u_h}).\vec{v_h} - \int_\Omega p_h(\nabla.\vec{v_h}) = \int_{\partial\Omega_N}\vec{s}.\vec{v_h}\ for\ all\ \vec{v_h} \in \mathbf{X}_0^h, \tag{9}$$

$$\int_\Omega q_h(\nabla.\vec{u_h}) = 0\ for\ all\ q_h \in M^h. \tag{10}$$

We see in the relations (9) and (10) that no derivative of p_h and q_h are used. It is sufficient that p_h and q_h are integrable. For $\vec{u_h}$ and $\vec{v_h}$, the integral of first derivative must exist. So we do not need the continuity of p_h and q_h in the weak formulation and that plays an important role in the element selection.
In the standard Galerkin method we define two basis functions, $\psi_i(x)$ for pressure and $\phi_i(x)$ for velocity. So the approximation for $\vec{u_h}$ and p_h is defined as

$$p_h = \sum_{j=1}^{m} p_j\psi_j(x), \tag{11}$$

and

$$\vec{u_h} = \sum_{j=1}^{n} u_{1j}\phi_{j1}(x) + u_{2j}\phi_{j2}(x) = \sum_{j=1}^{2n} u_j\phi_j(x), \tag{12}$$

where u_j is defined by $u_j = u_{1j}$, $j = 1, ..n$, $u_{j+n} = u_{2j}$, $j = 1, ...n$ and ϕ_j in the same way. In order to get the standard Galerkin formulation we substitute $\vec{v_i} = \phi_i(x)$, $q = \psi_i(x)$. Find p_h and $\vec{u_h}$, such that

$$\nu\int_\Omega\nabla\vec{u_h} : \nabla\phi_i + \int_\Omega(\vec{u_h}.\nabla\vec{u_h}).\phi_i - \int_\Omega p_h(\nabla.\phi_i) = \int_{\partial\Omega_N}\vec{s}.\phi_i\ for\ i = 1, ..2n, \tag{13}$$

$$\int_\Omega \psi_i(\nabla.\vec{u_h}) = 0\ for\ i = 1, .., m, \tag{14}$$

Formally the system of equations can be written as

$$Au + N(u) - B^T p = F \tag{15}$$

$$Bu = 0. \tag{16}$$

Where u denotes the vector of unknowns u_{1i} and u_{2i}, p denotes the vector of unknowns p_i. Au is the discretization of the viscous term and $N(u)$ the discretization of the nonlinear convective term, Bu denotes the discretization of of the divergence of u and $B^T p$ is the discretization of the gradient of p. The right hand side vector F contains all the contributions of the source term, the boundary integral as well as the contribution of the prescribed boundary conditions.
Since only linear systems of equations can be solved directly, equations (13,14) have to be linearized and combined with some iteration process. Commonly used schemes are Picard iteration and Newton iteration.

Picard iterations. In the Picard iteration method, the velocity in the previous step is substituted into the convective term. The convective term at the new level is given as

$$u^{k+1}.\nabla u^{k+1} \approx u^k.\nabla u^{k+1},$$

starting with initial guess $u^{(0)}$ for the velocity field, Picard's iteration constructs a sequence of approximate solutions $(u^{(k+1)}, p^{(k+1)})$ by solving a linear Oseen problem

$$-\nu\Delta u^{(k+1)} + (u^{(k)}.\nabla)u^{(k+1)} + \nabla p^{(k+1)} = f \text{ in } \Omega, \tag{17}$$

$$\nabla.u^{(k+1)} = 0 \text{ in } \Omega, \tag{18}$$

$k = 1, 2, \ldots.$ No initial pressure is required to be specified. Setting $u^{(k)} = 0$ correspond to the Stokes equations as given in (3) and (4).

Newton iterations. Newton's method is characterized by the fact that it is a quadratically converging process. Once it converges, it requires only few iterations. Suppose we write the solution at new level as the sum of the preceding level and a correction:

$$u^n = u^{n-1} + \delta u^{n-1},$$

If the nth iteration u^n is in the neighborhood of u , δu is small. The convective terms can be written as:

$$\begin{aligned} u^n.\nabla u^n &= (u^{n-1} + \delta u^{n-1}).\nabla(u^{n-1} + \delta u^{n-1}) \\ &= u^{n-1}.\nabla u^n + (u^n - u^{n-1}).\nabla(u^{n-1} + \delta u^{n-1}) \\ &= u^{n-1}.\nabla u^n + u^n.\nabla u^{n-1} - u^{n-1}.\nabla u^{n-1} + \delta u^{n-1}.\nabla\delta u^{n-1}. \end{aligned}$$

Neglecting the quadratic term in δu, the linearized form of (1)(2) become:

$$\nu\Delta u^n + u^n.\nabla u^{n-1} + u^{n-1}.\nabla u^n + \nabla p^n = f + u^{n-1}.\nabla u^{n-1}, \tag{19}$$

$$\nabla.u^n = 0. \tag{20}$$

Equations (19) and (20) are known as the Newton linearization of the Navier Stokes equations and continuity equation. The Stokes equations can be used as an initial guess. Since Newton iteration largely depends upon the initial guess, for high Reynolds number, the method does not converge due to a bad initial guess. In such a case few Picard iterations are suggested initially. Another good starting guess can be achieved by starting with a smaller Reynolds number, compute the solution and use this solution as an initial guess for large Reynolds number. This method is known as a continuation method.
After linearization the system can be again written as

$$Au + N(u^k)u + B^T p = F,$$

$$Bu = 0,$$

where u^k is the solution of the previous iteration.

Element selection conditions. Now that the problem of the nonlinear term is solved, the linear system arising from (1) and (2) can be written as

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{21}$$

where $F = A + N(u^k)$.

Another problem arises due to the zeros at the main diagonal of (21), which shows the absence of the pressure in the continuity equation. In (14) we see that the number of equations for velocity unknowns is determined by the pressure unknowns. If the number of pressure unknowns is larger than the number of velocity unknowns, the coefficient matrix in (21) becomes rank deficient, so we infer that number of pressure unknowns should never exceed the number of velocity unknowns irrespective of the grid size. To meet this criterion, the pressure must be approximated by interpolation polynomials that are at least one degree less than the polynomials for the velocity. But experiments show that this condition is not sufficient when the number of elements in a grid are small. However, it has been observed that even in case of an increase in number of elements, the matrix still remains singular. An exact conditions that elements must satisfy is known as Brezzi-Babuska condition (BB condition). The condition states that, for BB^T in (21) to be invertible it is necessary that
kernel(B^T)= 0, where B^T is nxm.
kernel(B^T)= 0 means that B^T has rank m, and is equivalent to requiring

$$\max_v (Bv, p) = \max_v (v, B^T p) > 0, \forall p. \tag{22}$$

The above relation in the framework of finite element method is

$$\max_{v \in V_h} \frac{(\nabla . v_h, q_h)}{\|v_h\|_{V_h} \|q_h\|_{Q_h}} > 0. \tag{23}$$

The above condition (23) allows the family of matrices to degenerate towards a singular system as $h \to 0$. The strict Brezzi Babushka condition ensures that BB^T not degenerates towards zero as h decreases. The modified form of (23) is given as

$$\inf_{q \in Q_h} \sup_{v \in V_h} \frac{(\nabla . v_h, q_h)}{\|v_h\|_{V_h} \|q_h\|_{Q_h}} \geq \gamma \geq 0. \tag{24}$$

It is very difficult to verify if the BB condition is satisfied or not. Fortin [3] has given a simple method to check the BB condition on a number of elements, which state that
an element satisfies BB condition, whenever given a continuous differentiable vector field u, one can explicitly build a discrete vector field $\tilde{u}$ such that

$$\int_\Omega \psi_i \ div \ \tilde{u} \ d\Omega = \int_\Omega \psi_i \ div \ u \ d\Omega \ for \ all \ basis \ functions \ \psi_i.$$

Note that all elements used in this report satisfies the BB condition. The elements used in a finite element discretization of the Navier Stokes equations are usually subdivided in two families, one having continuous pressure unknown (Taylor Hood family) and the Crouzeix Raviart family having a discontinuous pressure unknown. In Figure (1) and (2), some of the commonly used elements of these families in 2-D are shown. Both quadrilateral and triangular elements are used with different combinations of velocity and pressure polynomials. In the Crouzeix Raviart family the elements are characterized by a discontinuous pressure; discontinuous on element boundaries. For output purposes, these discontinuous pressures are averaged in vertices for all the adjoining elements. For details see [4].

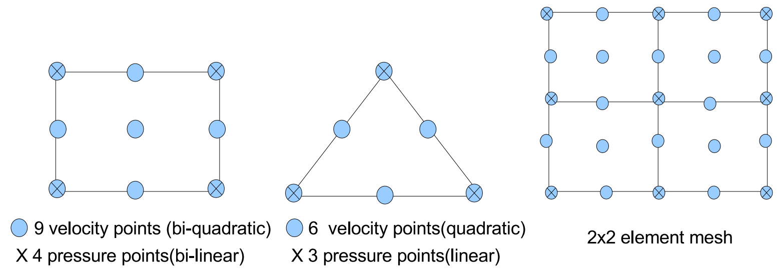
9 velocity points (bi-quadratic)
X 4 pressure points(bi-linear)
6 velocity points(quadratic)
X 3 pressure points(linear)
2x2 element mesh

FIGURE 1. Taylor Hood family elements $(Q2 - Q1)$, $(P2 - P1)$ elements and $(Q2 - Q1)$ grid

Another class of elements from the Taylor Hood family which satisfies the BB condition is known as mini-element, in which the velocity is defined by a bi-linear interpolation polynomial for the vertices with a bubble function at centroid and pressure is defined as a bi-linear polynomial. The bubble function is 1 in the centroid and zero on the node and therefore on the edges. It is necessary to prevent an overdetermined system of equations for the continuity equation. Since the bubble function is strictly local for an element,the centroid only contributes to the element matrix and vector for the specific element it is lying in. The rectangular and triangular mini elements are shown in Figure 3.

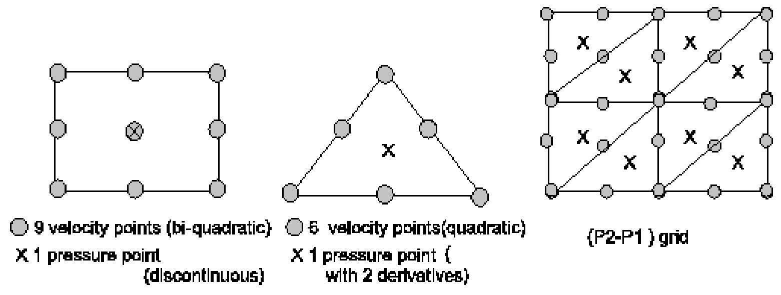
9 velocity points (bi-quadratic)
X 1 pressure point (discontinuous)
6 velocity points(quadratic)
X 1 pressure point (with 2 derivatives)
(P2-P1) grid

FIGURE 2. Crouzeix Raviart family elements $(Q2 - P0)$, $(P2^+ - P1)$ elements and $(P2^+ - P1)$ grid

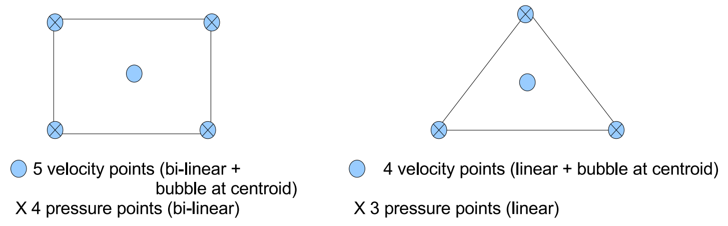
5 velocity points (bi-linear + bubble at centroid)
X 4 pressure points (bi-linear)
4 velocity points (linear + bubble at centroid)
X 3 pressure points (linear)

FIGURE 3. Taylor Hood family mini elements $Q_1^+ - Q_1$ element, $P_1^+ - P_1$ element

3. Solution Strategies

The linear system arising from the system of equations can be written in the form of

$$Ax = b$$

with A a nonsingular n by n matrix. This kind of systems can be solved in two ways, either a direct method such as a Gaussian elimination or an iterative method.

3.1. Direct method.

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

$$A = LU,$$

where L is a lower triangular matrix and U is an upper triangular matrix. We have to solve $LUx = b$, which can be done in the following steps
first solve $Ly = b$,
then solve $Ux = y$.
A direct method can be used when the matrix is dense. However, a sparse linear system with suitable sparsity structure is often solved with a direct method, because a direct method leads to more accurate solution and a fixed amount of work. For sparse matrices, sparseness can be used to reduce the computing time and memory during elimination process. An example of such kind of matrices is the band matrix in which nonzero elements are only on the main and some adjacent diagonals.

$$A = \begin{bmatrix} x & x & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & 0 & 0 & 0 \\ 0 & x & x & x & x & 0 & 0 \\ 0 & 0 & x & x & x & x & 0 \\ 0 & 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x & x \end{bmatrix} \tag{25}$$

In (25), matrix A is a band matrix with the lower bandwidth p, if $i > j+p \Rightarrow a_{ij} = 0$ and upper bandwidth q if $j > i+q \Rightarrow a_{ij} = 0$ and having bandwidth $p+q+1$. The system arising from finite element and finite difference methods are such that p and q are equal. Each entry within the band can be either zero or nonzero and all the elements outside the band are zero and remain zero during the elimination process, due to which L and U inherits the lower and upper bandwidth of A. The cost of the banded solution methods is governed by bandwidth, that is why these schemes may be inefficient for sparse matrices which contain a significant number of zeros inside the band. One alternative to the bandwidth strategy involves discarding all leading zeros in each row and column and storing only the profile of a matrix. The method is known as profile or envelope method.
Let A be square matrix, the lower envelope of A is the set of all the ordered pairs (i, j) such that $i > j$ and $a_{ik} \neq 0$ for $k \leq j$. The upper envelope of A is the set of ordered pairs (i, j) such that $i < j$ and $a_{kj} \neq 0$ for some $k \leq j$. Thus the upper envelope is the set of all the elements above the main diagonal excluding leading zeros in each column. If a matrix is symmetric and positive definite then $A = LL^T$, where L is the lower triangular matrix. This is known as the Cholesky factorization. In Cholesky factorization, L has the same envelope as A and we can save computer storage by employing a data structure that stores only the half band (lower or upper) of A and L can be stored over A. Both band and profile schemes depends on the order in which the equations and unknowns are numbered. We will discuss some of the renumbering schemes later, which minimize the envelope.
In a direct method, The LU factorization is the costly part of the computational process and the solution of the two steps is usually of minor cost. The elimination process fills the non-zero entries of a sparse matrix within a band or profile. So large number of entries has to be stored and the CPU time increases. Generally,

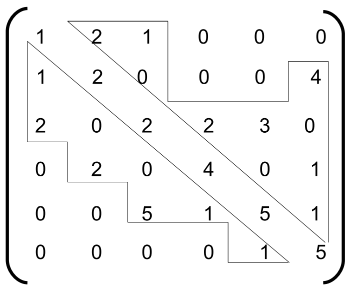

FIGURE 4. Profile of matrix

the system arising from the discretization of the finite element method has a sparse structure, which means that it contains a large number of zeros. The aim of a sparse direct solvers is to avoid doing operations on zero entries and therefore to try to minimize the number of fill-ins. We may save the computational cost and CPU time with an efficient reordering strategy which can be used to modify the structure of the matrix.

3.2. **Iterative methods.** Suppose we want to solve a linear system

$$Ax = b. \tag{26}$$

A is a non-singular square matrix and b is given. An iterative method constructs a sequence of vectors x^k, $k = 0, 1, ...,$ which is expected to converge towards x which is the solution of (26) and x_0 is given. The method is said to be convergent if $lim_{k\to\infty}\|x - x_k\| = 0$. Usually, the matrix A is splitted into two matrices

$$A = M - N.$$

The sequence x_k can be defined as

$$Mx_{k+1} = Nx_k + b. \tag{27}$$

Let $e_k = x - x_k$ is the error at the kth iteration. Then (27) can be written as

$$M(x - x_{k+1}) = N(x - x_k)$$

$$e_{k+1} = M^{-1}Ne_k$$

$$e_{k+1} = (M^{-1}N)^k e_0.$$

The method converges if $lim_{k\to\infty}(M^{-1}N)^k = 0$.

Theorem 1. *The iterative method (27) converges to $x = A^{-1}b$ if $\sigma(M^{-1}N) < 1$ where $\sigma(M^{-1}N) = max\{|\lambda|,$ where λ is spectrum of $M^{-1}N\}$, the set of eigenvalues of $M^{-1}N$ is said to be the spectrum of $M^{-1}N$.*

It is not easy to check the spectrum, since for most of the problems the eigenvalues of $(M^{-1}N)$ are not explicitly known. For more details see ([9], chapter 5). The variants of (27) are known as classical iterative methods. Gauss Seidel, Gauss Jacobi and SOR(successive over relaxation) are various types of classical methods. Some of the advantages of the iterative methods are:

- they need no modification of a given matrix, so no fill-in is generated and they do not need additional space for new elements which require additional time on inserting these elements into a complicated data structure,
- they need very little additional memory,
- they are usually faster than direct methods, especially when we do not need very good accuracy offered by direct method,
- they are easy to implement efficiently.

However, the iterative methods have some disadvantages too, we do not know the time needed to achieve the required accuracy. Moreover, sometimes one has problems with convergence when the required accuracy is high.
We will discuss another class of iterative method known as Krylov subspace methods.

3.3. Krylov subspace methods.

In the iterative method in (27), we replace $N = M - A$, then it can be written as

$$x_{k+1} = x_k + M^{-1}r_k, \tag{28}$$

where $r_k = b - Ax_k$ known as residual, if we start with x_0, the next steps can written as

$$x_1 = x_0 + (M^{-1}r_0)$$

$$x_2 = x_1 + (M^{-1}r_1)$$

replacing x_1 from previous step and $r_1 = b - Ax_1$, the above equation leads to

$$x_2 = x_0 + 2M^{-1}r_0 - M^{-1}AM^{-1}r_0$$

.
.
.
this implies that

$$x_k \in x_0 + span\{M^{-1}r_0, M^{-1}A(M^{-1}r_0), ..., (M^{-1}A)^{k-1}(M^{-1}r_0)\}$$

The subspace $K_k(A; r_0) := span\{r_0, Ar_0, ..., A^{k-1}r_0\}$ is called the Krylov subspace of dimension k corresponding to matrix A and initial residual r_0. It means that the Krylov subspace is spanned by the initial residual and by vectors formed by repeated multiplication of the initial residual and system matrix.
The Krylov subspace is constructed by basis $v_1, v_2, ..., v_j$. The basis is computed by the Arnoldi [7] algorithm. We start with $v_1 = r_0/\|r_0\|_2$, then compute Av_1, make it orthogonal to v_1 and normalize it, this gives v_2. The general procedure to form the orthonormal basis is as follows: assume we have an orthonormal basis $v_1, v_2, .., v_j$ for $K_j(A; r_0)$, this basis is expanded by computing $w = Av_j$ and orthonormalize this with respect to previous basis. The most commonly used algorithm is Arnoldi with modified Gram-Schmidt procedure as shown in Algorithm 1 [8]. Let the matrix V_j be given as

$$V_j = [v_1, v_2, ..., v_j] \in K_j$$

The columns of V_j are orthogonal to each other. It follows that

$$AV_{m-1} = V_m H_{m,m-1}.$$

The m by $m-1$ matrix $H_{m,m-1}$ is upper Hessenberg, and its elements $h_{i,j}$ are defined by the Arnoldi algorithm [7].

The Arnoldi algorithm is composed of a matrix vector products, inner products and vector updates. If A is symmetric, then $H_{m-1,m-1} = V_{m-1}^T AV_{m-1}$ is also symmetric and tridiagonal and leads to a three term recurrence in the Arnoldi process. Each new vector has to be orthogonalized with respect to previous two vectors only. The algorithm is known as the Lanczos algorithm. Krylov subspace methods are developed on the bases of these algorithms. We will discuss some of the Krylov subspace methods which are used in numerical simulations.

Algorithm 1 Arnoldi algorithm with modified Gram-Schmidt procedure

```
$v_1 = r_0/\|r_0\|_2$;
For $j = 1$ to $m-1$
    $w = Av_j$;
    For $i = 1$ to $j$,
        $h_{i,j} = v_i^T w$;
        $w = w - h_{i,j} v_i$;
    end
    $h_{j+1} = \|w\|_2$;
    $v_{j+1} = w/h_{j+1,j}$;
```

3.3.1. ***GMRES.*** The generalized minimal residual algorithm is developed by Saad and Schultz [10]. The method is based on long recurrences and satisfies optimality property. The method is used for nonsymmetric nonsingular matrices. GMRES is based on a modified Gram-Schmidt orthonormalization procedure and uses a restart to control storage requirements. From the algorithm, it is clear that Arnoldi algo-

Algorithm 2 GMRES algorithm

```
1.  Compute $r_0 = b - Ax_0$, $\beta := \|r_0\|_2$, and $v_1 = r_0/\beta$
2.  For $j = 1$ to $m$
3.      Compute $w = Av_j$;
4.      For $i = 1$ to $j$
5.          $h_{ij} := (w_j, v_i)$
6.          $w_j := w_j - h_{ij} v_i$
7.      End
8.      $h_{j+1,j} = \|w_j\|_2$. if $h_{j+1,j} = 0$ set $m := j$ and exit for
9.      $v_{j+1} = w_j/h_{j+1,j}$
10. End
11. Define the $(m+1)$x$m$ Hessenburg matrix $\bar{H_m} = \{h_{ij}\}_{1\leq i\leq m+1, 1\leq j\leq m}$
12. Compute $y_m$, the minimizer of $\|\beta e_1 - \bar{H_m} y\|_2$, and $x_m = x_0 + V_m y_m$
```

rithm is followed by a minimum least square problem.

$$J(y) = \|b - Ax\|_2 = \|b - A(x_0 + V_m y\|_2$$

by using $r_0 = b - Ax$, $AV_m = V_{m+1}\bar{H_m}$, $e_1 = [1, 0, ..., 0]^T$the above relation leads to minimization of

$$J(y) = \|\beta e_1 - \bar{H_m} y\|_2.$$

GMRES is a stable method and no breakdown occurs, if $h_{j+1,j} = 0$ then $x_m = x$ reached the solution. It can be seen that work per iteration and memory requirements increase for an increasing number of iterations. In order to avoid the problem of excessive storage requirement and computational costs for the orthogonalization, GMRES is usually restarted after m iterations which uses the last iteration as starting vector for next restart. The restarted GMRES is denoted as GMRES(m). Now the problem arises which are suitable choices of m. There is no rule for the choice of m. A disadvantage in this approach is that the convergence behavior in many cases seems to depend quite critically on the choice of m. The property of superlinear convergence is lost by throwing away all the previous information of the Krylov subspace. If no restart is used, GMRES (like any orthogonalizing Krylov subspace method) will converge in no more than N steps(assuming exact arithmetic). For more details on the GMRES convergence see [15].

Theorem 2. *Suppose that A is diagonalizable so that $A = XDX^{-1}$ and let*

$$\epsilon^{(m)} = \min_{\substack{p \in p_m \\ p(0)=1}} \max_{\lambda_i \in \sigma} |p(\lambda_i)|$$

then the residual norm of the m-th iterate satisfies

$$\|r_m\|_2 \leq K(X)\epsilon^{(m)}\|r_0\|_2$$

where $K(X) = \|X\|_2\|X^{-1}\|_2$. If furthermore all eigenvalues are enclosed in a circle centered at $C \in \mathbb{R}$ with $C > 0$ and having radius R with $C > R$, then $\epsilon^{(m)} \leq (\frac{R}{C})^m$.

3.3.2. ***GMRESR***. This method is a variant of GMRES developed by Vuik and van der Vorst [11]. The idea is that the GMRES method can be effectively combined with other iterative schemes. The outer iteration steps are performed by GCR, while the inner iteration steps can be performed by GMRES or with another iterative method. In GMRESR, inner iterations are performed by GMRES. For $m = 0$,

Algorithm 3 GMRESR algorithm

```
1.  $x_0$ is an initial guess and $r_o = b - Ax_0$
2.  For $j = 1, 2, 3...$
3.      $s_i = P_{m,i-1}(A)r_{i-1}$,
        $s_i$ be the approximate solution of $As = r_{i-1}$
        obtained after $m$ steps of an iterative method
4.      $v_i = As_i$
5.      For $j = 1$ to $i - 1$
6.          $\alpha = (v_i, v_j)$,
7.          $v_i = v_i - \alpha v_j$,$s_i = s_i - \alpha s_j$,
8.      End
9.      $v_i = v_i/\|v_i\|_2$,$s_i = s_i/\|v_i\|_2$
10.     $x_i = x_{i-1} + (r_{i-1}, v_i)s_i$;
11.     $r_i = r_{i-1} - (r_{i-1}, v_i)v_i$;
12. End
```

we get GCR and for $m \to \infty$ we get GMRES. The amount of work and required memory for GMRESR is much less than GMRES. The choice of m in GMRESR is not critical. The proper choice of m and amount of work has been discussed in [11]. In some cases, when the iterative method is close to the solution (satisfy stopping criteria), the m inner iterations of GMRES at that point will lead to high accuracy which is not required at that point. So it is never necessary to solve the inner iterations more accurately than the outer one [11].

3.3.3. ***Bi-CGSTAB***. Bi-CGSTAB [14] is a variant of the Bi conjugate gradient(Bi-CG) [12] algorithm. If the matrix A is symmetric and positive definite then CG algorithm converges to the approximate solution. The CG method is based on Lanczos algorithm. For nonsymmetric matrices Bi-CG algorithm was introduced which is based on Lanczos biorthogonalization. The algorithm not only solves the original system $Ax = b$ but also a linear system $A^T x^* = b$. In the Bi-CG method, the residual vector can be written in as $r_j = \phi_j(A)r_0$ and $\bar{r_j} = \phi_j(A^T)\bar{r_0}$, Sonneveld [13] observed that we can also construct the vectors $r_j = \phi_j^2(A)r_0$, using only the latter form of the innerproduct for recovering the bi-conjugate gradients parameters(which implicitly define the polynomial ϕ_j). With the CGS method, the formation of vector $\bar{r_j}$ and multiplication with A^T can be avoided. However, CGS shows irregular convergence behavior in some cases. To remedy this difficulty Bi-CGSTAB(Bi-conjugate gradient stabilized) is developed by van der Vorst [14]. Bi-CGSTAB produces iterates with residual vectors of the form

$$r_j = \psi_j(A)\phi_j(A)r_0,$$

ψ_j is the new polynomial defined recursively at each step for stabilizing or smoothing the convergence. The advantage of this method is a short recurrence. It is always

Algorithm 4 Bi-CGSTAB algorithm

```
1.  $x_0$ is an initial guess and $r_o = b - Ax_0$
2.  Choose $\bar{r_0}$ is an arbitrary vector, for example $\bar{r_0}$
3.  $\rho_{-1} = \alpha_{-1} = \omega_{-1} = 1$
4.  $v_{-1} = p_{-1} = 0$
5.  For $i = 0, 1, 2, 3...$
6.      $\rho_i = (\bar{r_0}, r_0); \beta_{i-1} = (\rho_i/\rho_{i-1})(\alpha_{i-1}/\omega_{i-1})$
7.      $p_i = r_i + \beta_{i-1}(p_{i-1} - \omega_{i-1}v_{i-1})$
8.      $v_i = Ap_i$
9.      $\alpha_i = \rho_i/(\bar{r_0}, v_i)$
10.     $s = r_i - \alpha_i v_i$
11.     if $\|s\|$ is small enough then $x_{i+1} = x_i + \alpha_i p_i$, exit For
12.     $t = As$
13.     $w_i = (t, s)/(t, t)$
14.     $x_{i+1} = x_i + \alpha_i p_i + w_i s$
15.     if $x_{i+1}$ is accurate enough then exit For
16.     $r_{i+1} = s - w_i t$
17. End For
```

necessary to compare the norm of the updated residual to exact residual as small changes in algorithm can lead to instabilities.

4. Preconditioning

Preconditioning is a technique used in iterative methods for solving large linear systems, which influence an iterative method such that it converges rapidly. Instead of solving a system $Ax = b$, one solves a system $M^{-1}Ax = M^{-1}b$, where M is the preconditioner. A good preconditioner must have a variety of properties. First, the preconditioned system should converge quickly. This generally means that $M^{-1}A$ has a small condition number. Secondly, it should be easy to solve systems of the form $My = z$. The construction of the preconditioner should be efficient in both time and space. In Krylov subspace methods, the convergence depends on the eigenvlaues distribution, so a preconditoned system must have a favorable spectrum (clustered around 1).

Some of the above properties can be acheived by reducing the profile or bandwith of the matrix. In order to reduce the band or profile, several algorithms for reordering rows and columns exist, that minimizes the amount of fill-in and thus computation time. We will use some of these methods in combination with reordering of unknowns. The scheme will work for a direct method and will be used in the preconditoner for an iterative method to enhance the convergence.

Most of the preconditioners are application-specific which exploits the problem insight. We will discuss here such kind of preconditioner designed for Navier Stokes equations, which we will use with GMRES(m), GMRESR and BiCGSTAB.

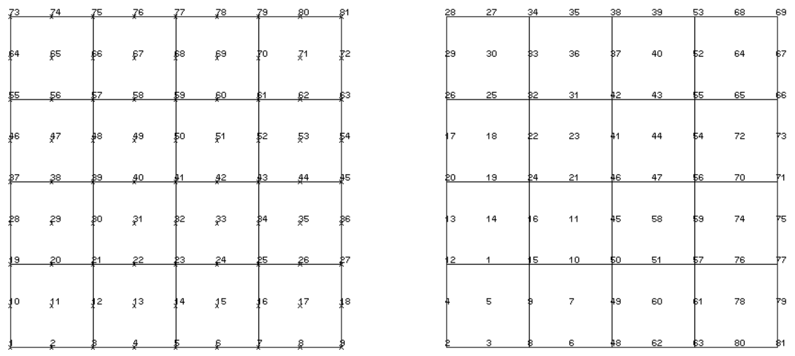

FIGURE 5. (a)-Lexicographic mesh numbering, (b)-Sloan mesh renumbering

unknowns ordered per nodal points
$u_1v_1p_1\ u_2v_2\ u_3v_3p_3, \ldots., u_5v_5p_5\ u_6v_6\ u_7v_7\ u_8v_8, \ldots.u_{25}v_{25}p_{25}$
unknowns ordered p-Last
$u_1v_1\ u_2v_2\ u_3v_3 \ldots\ u_{25}v_{25}\ p_1\ p_3\ p_5\ p_{11}\ p_{13}\ p_{15}..p_{25}$
unknowns ordered p-last per level
$u_1v_1\ u_2v_2 \ldots\ u_{15}v_{15}\ p_1\ p_3 \ldots p_{15}\ u_{16}v_{16}\ u_{17}v_{17} \ldots u_{25}v_{25}\ p_{21}\ p_{23}..\ p_{25}$

TABLE 1. Different ordering schemes of unknowns for mesh shown in Figure (7)

4.1. **ILU Preconditioner with mesh renumbering and unknowns reordering.** From the discretization of the Stokes or Navier Stokes equation, we come up with a system of equations that can be written as:

$$\begin{bmatrix} A & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{29}$$

In the case of Navier Stokes, A denotes the discretization of the stress tensor and the linearized convective terms (in Stokes the convective term is absent), B^Tp is the discretization of the pressure gradient and $Bu = 0$ is the discretization of the continuity equation. The vectors u and p represent velocity and pressure unknowns, respectively.

In the case of a direct solver, the ordering of the unknowns suggested in (29), that is, first all velocity unknowns and then all pressure unknowns appears to be very inefficient, since the corresponding profile is very large. A much smaller profile can be achieved if the velocity and pressure unknowns are intermixed, which can be obtained by ordering all the unknowns per point. Due to boundary conditions it may be possible that the first diagonal element in the coefficient matrix is zero, due to which the LU decomposition fails. Partial pivoting can change the profile or bandwidth of the matrix so we do not use this technique. In fact one would like to have some a priori numbering of nodal points and unknowns that prevents the presence of a zero diagonal element during the elimination process and produces an optimal profile. This numbering must be such that the first unknowns are velocities independent of the type of boundary conditions. For this purpose the grid points are renumbered using various numbering schemes. We will discuss (see Appendix A) two renumbering schemes here.

(1) Sloan's algorithm [16].
(2) Cuthill McKee's algorithm [17].

A small profile can be achieved by the Cuthill McKee or Sloan renumbering algorithm. Figure 5 shows the Sloan's renumbered grid. Figure 6 reflects how the grid is numbered levelwise using the Cuthill McKee numbering scheme. The sparsity pattern of the coefficient matrix with Sloan and Cuthill McKee numbering using p-last per level reordering is shown in Figure 8) and 9. Both methods produce a nearly optimal profile. The new numbering is known as renumbering per level. The idea is as follows, in each step of the algorithm a new level of elements is created, by a new set of neighbors. In this new level we reorder the unknowns such that the velocity unknowns are followed by pressure unknowns. With this the local bandwidth is hardly effected. We call this: p-last per level. Usually, the first level contains more nodal points than the other levels. For example the levels defined for 8x8 $Q2-Q1$ grid is given as

```
levels= [50 13 18],
% first  level= 50 nodal points,
% second level= 13 nodal points,
% third  level= 18 nodal points.
```

In p-last, all the velocity unknowns are followed by pressure unknowns in the grid. Both orderings are shown in Table 1.

Using renumbering of the unknowns(preferably per level), in combination with ILU preconditioning increases the convergence of the non-symmetric CG-type methods like BiCGSTAB and GMRES. In our method, first the mesh grid points are renumbered then the unknowns per grid point are reordered by the p-last, or the p-last per level reordering method. After reordering, the ILU decomposition of the coefficient matrix is constructed and used as preconditioner [18]. Preconditioned Krylov subspace methods are then used to approximate the solution.

For the ILU decomposition, the coefficient in matrix (29) is decomposed into the following matrices: L a lower triangular matrix, D a diagonal matrix, U an upper triangular matrix, where $diag(L) = diag(U) = D$. The decomposition is made such that $\hat{L}\hat{U} \approx LD^{-1}U$ and the following rules are used:

$l_{i,j} = 0$ if $a_{ij} = 0$ for $i > j$,

$u_{i,j} = 0$ if $a_{ij} = 0$ for $i < j$,

$(\hat{L}\hat{U})_{i,j} = a_{i,j}$ if $a_{ij} \neq 0$.

The system of equations in (29) can also be represented as:

$$M \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} A & B^T \\ -B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix} \tag{30}$$

The rate of convergence of Krylov subspace iterative methods depends on the spectrum of the coefficient matrix. In the case of a Stokes problem the spectrum of (29) contains positive and negative eigenvalues, whereas (30) has a spectrum containing complex eigenvalues, but the real parts of all eigenvalues are positive. In general, a preconditioner is constructed such that the eigenvalues of the preconditioned matrix are clustered. If a matrix has eigenvalues with positive part, no problem is expected. However, in other cases when the matrix has positive and negative eigenvalues, difficulties may occur, because it is possible that negative eigenvalues become eigenvalues of the preconditioned matrix very close to zero, which leads to bad convergence [18]. In our case such problems have not been seen. However, although (29) is symmetric, the preconditioner can not be used for MINRES or SYMMLQ [19] algorithms, since the preconditioner is not positive definite which is the requirement for these two methods. The diagonal elements of the pressure part of the preconditioner appear to be negative.

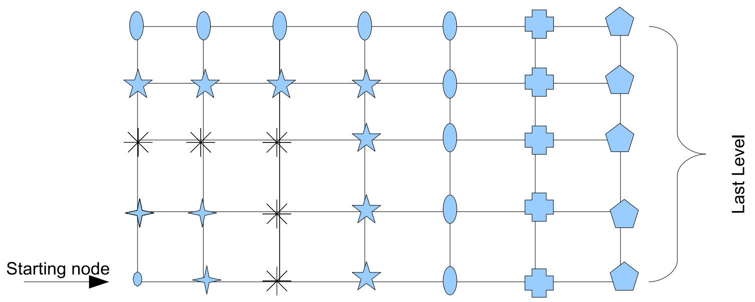

FIGURE 6. Cuthill McKee renumbering

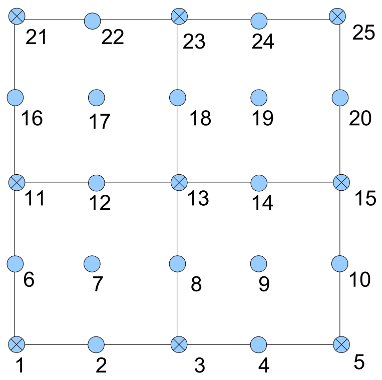

FIGURE 7. A (2x2) Q2-Q1 mesh

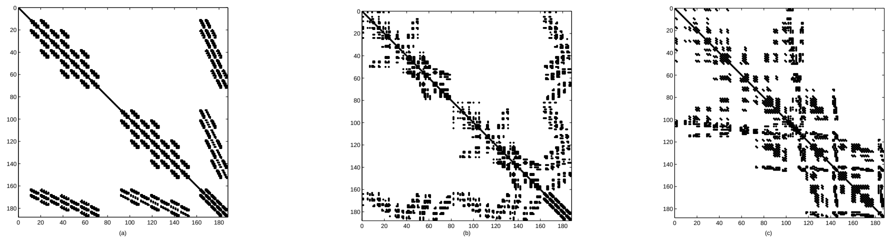

FIGURE 8. Sparsity pattern of Stokes matrix with (a)-Lexicographic numbering, (b)-Sloan renumbering with p-last reordering (c) Sloan renumbering with P-last per level reordering

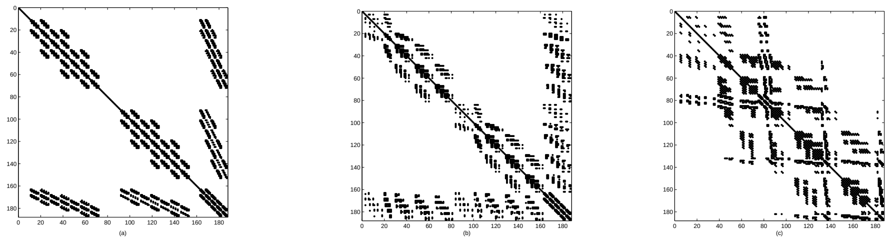

FIGURE 9. Sparsity pattern of Stokes matrix with (a)-Lexicographic numbering, (b)-Cuthill McKee renumbering with p-last reordering (c) Cuthill McKee renumbering with P-last per level reordering

5. Numerical Experiments

The Stokes and Navier Stokes problems are solved in two domains

(1) The Poiseuille channel flow in a square domain $(-1,1)^2$ with a parabolic inflow boundary condition and natural outflow condition having the analytic solution: $u_x = 1-y^2;\ u_y = 0;\ p = 2\nu x$. In the case of Stokes flow $\nu = 1$.
(2) The L-shaped domain $(-1,L)$x$(-1,1)$ known as backward facing step shown in Figure 10. 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 outflow which automatically sets the mean outflow pressure to zero.

The streamline plot for Stokes and Navier Stokes problems in the channel and backward facing step are shown in Figure 11 and Figure 12.

In the case of Stokes problem as can be seen from Table 2, a direct method gives the solution faster than iterative methods with the ILU preconditioner for small problems. However, memory requirements for the direct method are large as compared to the iterative solution methods [18] and at a certain point the time required to solve the Stokes equations by a direct method increases considerably as the mesh size increases. The mesh node numbering is carried out by a default renumbering scheme(Sloan) used in SEPRAN. The convergence of an iterative method depends on the mesh numbering, reordering of the unknowns and the ILU preconditioner. In absence of anyone of these, the iterative solver may not converge fast or may even diverge. From Table 2 and 3, it is clear that the p-last per level reordering gives much faster results than the p-last ordering both in iterations and time. The time taken by the convergence of BiCGSTAB and GMRESR is less than GMRES(m). Also BiCGSTAB uses less iterations as compared to GMRES(m). Note that one iteration of BiCGSTAB costs two matrix vector multiplications. The Preconditioned GMRES does not converge with increased number of grid points. If the number of iterations is small and a matrix vector product is expensive(which means a large number of non zero elements per row) then GMRES is the best method [18].

The effect of reordering the mesh by the Sloan or the Cuthill McKee algorithm is shown in Table 4. The Sloan reordering gives faster convergence than Cuthill McKee if the equation is discretized by the $Q2-Q1$(velocity defined as bi-quadratic field and pressure defined as bi-linear per element) or $Q2-P1$ (velocity bi-quadratic and pressure and its gradients discontinuous at element boundaries). But in the case of $\overset{+}{Q}1-Q1$(velocity and pressure are continuous and velocity at centroid is

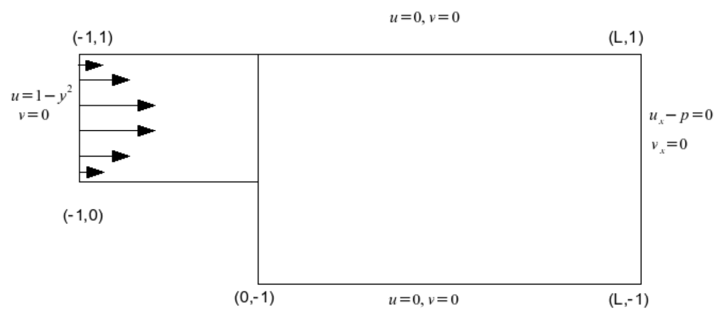

Figure 10. Backward facing step or L-shaped domain

eliminated at element level, thus reducing number of unknowns) discretization, the Cuthill McKee renumbering gives much better convergence results than the Sloan renumbering. In Table 5, the time elapsed in solving Stokes problem with a direct solver shows that it has also the same kind of results for both kind of renumbering schemes for various discretizations. In the case of the Navier Stokes problem, the number of inner and outer iterations increases with the increase in the Reynolds number and in grid points. The number of iterations shown in Table 6 and 7 are the total number of inner iterations taken by the iterative solver in the approximation of the linear problem. The results of Table 7 are also done with the ILU preconditioner with Sloan numbering and p-last per level reordering.

The non-linear problem can be linearized by two methods, Picard's method or Newton's method. As is well known, Newton's method is quadratically convergent in some cases, whereas Picard iteration has only a linear convergence rate. On the other hand, Newton's method is convergent only if the initial iterate is close enough to the solution. Also the cost of solving linear problems for non-linear iterations tend to be higher when Newton's method is used than for Picard iteration. So for Newton's method, the cost and convergence both depends upon the initial iterate and that can be improved by performing a few initial steps with Picard and then switch to Newton's method to get faster convergence. From Table 8, it is clear that an increase in Reynolds number increases the number of inner and outer iterations for both methods. One Picard step followed by Newton's method gives faster convergence results than the Picard method, but at high Reynolds number, because of a bad initial estimate, Newton's method diverges, so it is good to use only Picard iterations after the Stokes equations at high Reynolds number or few Picard iterations followed by Newton iterations. Though Picard method converges in more outer iterations, the average inner iteration required by Picard's method is less than Newton's method. Table 9 shows a very interesting result, the solution of the linearized problem with a direct solver has no impact on the outer iterations. Comparing the results of Table 8 with 9, the direct solver seems more expensive than the iterative solver. From these results we can conclude that it is not necessary to solve the linearized system to high accuracy, because the results from the direct solver reveals that the exact solution of linearized problem has no impact on the convergence behavior of both Picard and Newton iteration schemes. So the accuracy of $eps = 1e-2$ will be enough for the inner iterations if an iterative solver is used.

The effect of grid stretching is shown in Table 10. We increased the number of elements in one direction only, so that the aspect ratio of the element increases. For both kind of reordering schemes the number of iterations increases, but there is a clear difference between Sloan and Cuthill McKee renumbering. Cuthill McKee may sometimes diverge, while Sloan in combination with p-last per level converges always. Furthermore, we see a sharp dip(minimum) in the number of iterations for 64x24 grid. For this behavior we do not have an explanation. This observation has been found for cells with an 8:3 ratio in the backward facing step.

The Stokes and Navier Stokes equations are also solved in a 3-D backward facing step. A tri-linear and a tri-quadratic hexahedron elements from the Taylor Hood family are shown in Figure 13. In the 3-D Taylor Hood family, velocity is defined as tri-quadratic field and pressure as tri-linear field per element denoted as k2-k1. In the 3-D Crouzeix Raviart family, the pressure and gradients of pressure are defined in the centroid of an element denoted as k2-k0. Results with k2-k1 elements for Stokes problem are shown in Table 11, a 3-D Cuthill McKee numbering shows faster convergence than Sloan numbering with both p-last and p-last per level reordering schemes. A Stokes problem is also solved with a direct solver in a 8x8x8 grid with p-last per level ordering, it gives the solution in 20 seconds with Sloan numbering and 52 seconds with Cuthill McKee numbering consuming a huge amount of memory. In the k2-k0 elements, Sloan renumbering gives faster convergence than Cuthill McKee for both kind of reordering. Results given in Table 12 suggest that the Krylov subspace method converges in the same number of iterations with both orderings for Cuthill McKee renumbering. Table 13 shows that the results obtained from the solution of Navier Stokes problem, this reveals that the preconditioned Krylov methods have the same convergence behavior for different Reynolds number and grid size, as we have observed in the Stokes problem for various 3-D elements.

In Table 14 and 15, some fill-in and lumping is introduced in the preconditioner for a stretched grid. This shows improved convergence of an iterative method. However, the CPU time and memory usage increases. So it suggests that fill-in and lumping should only be used when the iterative method is not converging to the solution. In ILU(2) , all unknowns in the neighbour points of the unknown in a row matrix are also supposed to be the part of the nonzero structure, so the number of iterations decreases at the cost of extra memory and time per iteration. We have extended the continuity equation with a factor ϵp, which discretized form is equal to ϵM_p. ϵ is a small number. So the incompressibility condition is violated by a small amount, with hardly no influence on the solution. In the case of a $Q2-P1$ discretization for a certain value of ϵ, the convergence improves. However, it is very difficult to find suitable value of ϵ for better convergence.

In Table 16, we compare our preconditioner(p-last per level with Sloan renumbering) with the preconditioners implemented in IFISS (PDC- pressure convection diffusion [20], [21], LSC -least squares commutator [22], [23]) for Navier Stokes equations. Note that IFISS uses BICGSTAB(ℓ) [24] and GMRES (with modifications from Kelly [25]) for their preconditioners. Whereas in our report, we have used GMRES(20) and BICGSTAB. The number of iterations taken by an iterative solver in the last step of the outer loop are shown in Table 16. Our preconditoner seems more efficient in CPU time than both PDC and LSC preconditioners.

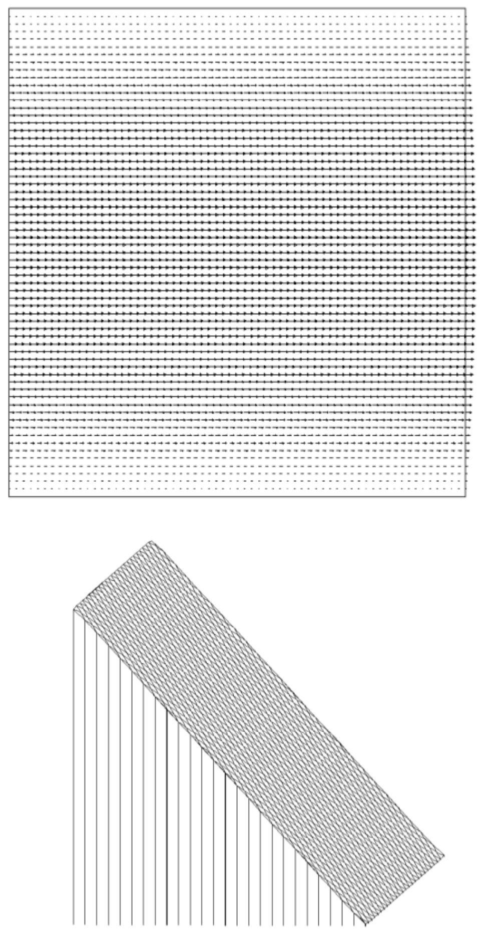

Figure 11. (a)-Channel flow, (b)- 3-D plot for pressure in channel flow

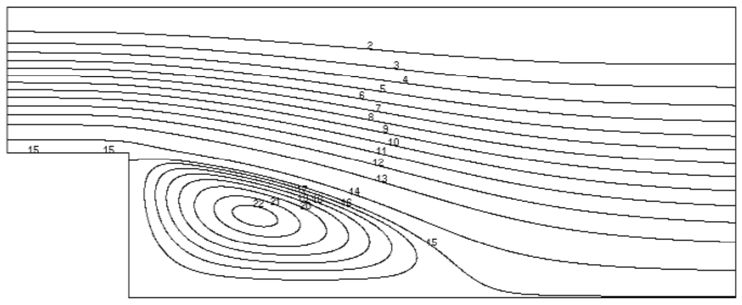

Figure 12. Flow in Backward facing step or L-shaped domain with $Re = 250$

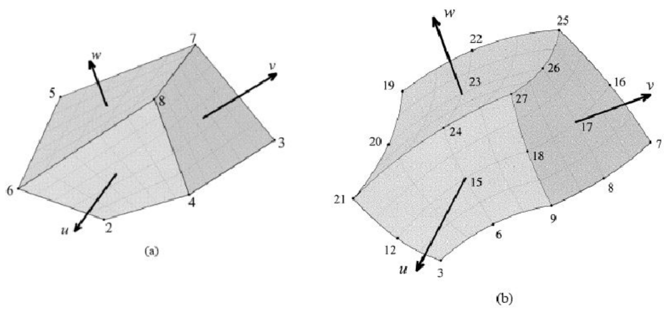

FIGURE 13. Generalized hexahedron (a) Tri-linear (b) Tri-Quadratic

Solver.	Precon.	Renumber	it/sec(16x16)	it/sec(32x32)	it/sec(64x64)
Direct	-	p-last-level	-/0.072	-/0.64	-/10.32
GMRES(20)	ILU	p-last	84/0.140	226/1.140	579/12.98
GMRES(20)	ILU	p-last-level	59/0.136	163/0.840	450/10.07
GMRESR	ILU	p-last	13/0.148	25/0.824	62/7.57
GMRESR	ILU	p-last-level	11/0.108	20/0.652	39/5.01
BiCGSTAB	ILU	p-last	36/0.112	92/0.912	224/7.34
BiCGSTAB	ILU	p-last-level	30/0.108	64/0.656	145/4.96

TABLE 2. Solution of Stokes Problems with $Q_2 - P_1$ discretization in square domain with $eps = 1e - 6$

Solver.	Precon.	Renumber	it/sec(16x16)	it/sec(32x32)	it/sec(64x64)
Direct	-	p-last-level	-/0.052	-/0.55	-/9.7
GMRES(20)	ILU	p-last	58/0.104	201/0.98	708/14
GMRES(20)	ILU	p-last-level	41/0.08	99/0.55	362/7.33
GMRESR	ILU	p-last	12/0.1	26/0.764	58/6.42
GMRESR	ILU	p-last-level	10/0.09	18/0.57	37/4.15
BiCGSTAB	ILU	p-last	39/0.095	74/0.648	190/5.95
BiCGSTAB	ILU	p-last-level	24/0.092	49/0.48	118/3.768

TABLE 3. Solution of Stokes Problems with $Q_2 - Q_1$ discretization in a square domain $eps = 1e - 6$

Grid	$Q2 - Q1$		$Q2 - P1$		$\overset{+}{Q}1 - Q1$	
-	Sloan	Cuthill-McKee	Sloan	Cuthill-McKee	Sloan	Cuthill-McKee
-	Iter.	Iter.	Iter.	Iter.	Iter.	Iter.
8x24	9	15	29	97	17	10
16x48	22	32	40	288	35	18
32x96	59	65	73	1300	75	37

TABLE 4. Effect of mesh renumbering on convergence of BiCGSTAB for various discretizations in backward facing step Stokes problem with P-last per level and $eps = 1e - 6$

Grid	$Q2-Q1$		$Q2-P1$		$\overset{+}{Q}1-Q1$	
-	Sloan	Cuthill McKee	Sloan	Cuthill McKee	Sloan	Cuthill McKee
-	Time(sec)	Time(sec)	Time(sec)	Time(sec)	Time(sec)	Time(sec)
8x24	0.03	0.06	0.04	0.09	0.06	0.04
16x48	0.25	0.56	0.33	1.25	0.57	0.19
32x96	4	11	5.6	27.8	7.17	2.78

Table 5. Effect of mesh renumbering on a direct solver for various discretization in backward facing step Stokes problem using p-last-level ordering

Reynolds no.	GMRES(20)	GMRESR	BiCGSTAB
-	iterations/time(sec)	iterations/time(sec)	iterations/time(sec)
$Re=50$	323/1.30	69/1.46	139/1.09
$Re=150$	372/1.48	77/1.60	165/1.22
$Re=250$	578/2.21	114/2.31	246/1.77

Table 6. Effect on iterations with varying Reynolds number for Navier Stokes backward facing step problem(16x48) with Q_2-Q_1 discretization with $eps=1e-3$ using p-last-level reordering

it-solver	$Re=10$			$Re=100$		
Type	8x24	16x48	32x96	8x24	16x38	32x96
-	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)
GMRES(20)	92/0.17	276/1.1	1505/18.83	128/0.21	366/1.46	1639/21
GMRESR	25/0.18	59/1.23	132/9.97	36/0.23	74/1.58	164/12.63
BiCGSTAB	54/0.14	130/0.94	350/8	81/0.21	162/1.24	431/10.21

Table 7. Effect of grid increase and Reynolds number on iterations for Navier Stokes backward facing step problem with $eps=1e-3$ using p-last-level reordering

$eps=1e-2$	Picard's Method			Newton's method		
Reynolds	Out. iter	in. iter.	time(sec)	out. iter	in. iter	time(sec)
50	7	106	1.04	5	73	0.75
100	9	135	1.28	5	80	0.77
150	10	148	1.46	6	104	0.968
200	12	163	1.63	7	126	1.124
250	14	188	1.86	7	145	1.224
400	28	352	5.5	-	-	-

Table 8. The effect of Reynolds number on non-linear iterations in Navier Stokes backward facing step problem with BiCGSTAB using p-last-level reordering for Q2-Q1(16x48) grid

	Picard's Method		Newton's method	
Reynolds	Out. iter	time(sec)	out. iter	time(sec)
50	7	1.5	5	1.124
100	9	1.9	5	1.132
200	12	2.6	7	1.55
400	28	5.9	-	-

Table 9. The number of outer iterations for various Reynolds number in Navier Stokes backward facing step problem with a direct Solver using p-last-level reordering for Q2-Q1(16x48) grid

$eps = 1e-4$	Q2-Q1				Q2-P1			
Grid	Sloan (iter.)		Cuthill McKee(iter.)		Sloan(iter.)		Cuthill McKee(iter.)	
	p-last	p-last-level	p-last	p-last-level	p-last	p-last-level	p-last	p-last-level
8x24	22	13	25	12	36	23	135	149
16x24	44	30	50	39	55	47	226	269
32x24	538	226	113	92	387	339	277	277
64x24	-	40	321	135	-	58	>3000	2757
128x24	-	265	1116	>3000	-	278	-	-

Table 10. Effect of stretched grid on convergence of preconditioned BiCGSTAB method in Stokes backward facing step problem

	Sloan numbering		Cuthill McKee numbering	
BiCGSTAB, $eps = 1e-4$				
Grid	p-last(it./time(s))	p-last-level(it./time(s))	p-last(it./time(s))	p-last-level(it./time(s))
8x8x8	39(2.5)	38(2.46)	33(1.76)	30(1.72)
8x8x16	40(6.2)	36(6)	36(4.8)	24(4)
12x12x12	85(16)	89(17)	67(10.3)	54(9)

Table 11. 3-D Stokes backward facing step problem using k2-k1(Taylor Hood family) elements

	Sloan numbering		Cuthill McKee numbering	
BiCGSTAB, $eps = 1e-4$				
Grid	p-last(it./time(s))	p-last-level(it./time(s))	p-last(it./time(s))	p-last-level(it./time(s))
8x8x8	56(2.9)	52(2.8)	66(2.6)	66(2.6)
8x8x16	51(6.9)	47(6.6)	60(6.2)	60(6.2)
12x12x12	91(16.7)	87(16.3)	124(15.2)	123(15.2)

Table 12. 3-D Stokes backward facing step problem using k2-k0(Crouzeix Raviart family) elements

Picard's method	k2-k1		k2-k0	
Reynolds number = 75, GMRES, $eps = 1e-2$				
Grid	Sloan(iter.)	Cuthill McKee(iter.)	Sloan(iter.)	Cuthill McKee(iter.)
8x8x8	294	179	578	744
8x8x16	230	127	505	667
Reynolds number = 50, BiCGSTAB, $eps = 1e-2$				
8x8x8	149	100	254	323
12x12x12	322	163	359	444
16x16x16	718	257	486	621
16x16x32	371	194	450	977

Table 13. Accumulated inner iterations for 3-D Navier Stokes backward facing step problem with p-last-level reordering

$Q2-P1$	Sloan renumbering, $eps = 1e-4$							
Grid	p-last				p-last-level			
	ILU(0)	ILU(2)	Lumped	ILU(2),Lumped	ILU(0)	ILU(2)	Lumped	ILU(2),Lumped
8x24	36(0.8)	14(1.5)	41(.8)	17(1.5)	23(.7)	8(1.2)	27(.8)	8(1.2)
16x24	55(1.6)	27(3.2)	71(1.7)	33(3.2)	47(1.7)	16(2.7)	52(1.7)	19(2.6)
32x24	364(4.9)	138(7.8)	295(4.7)	89(7.4)	159(4.7)	72(6.3)	456(5.3)	54(6.2)
64x24	-	-	>3000(*1)	237(19)*2	58(7.2)	18(11.2)	217(8.7)	66(12.7)
128x24	-	-	-	780(60)*3	293(19.3)*4	-	808(29)	224(29)
	Cuthill McKee renumbering, $eps = 1e-4$							
8x24	158(0.9)	14(2.0)	140(0.9)	17(2.0)	148(0.9)	7(2.05)	175(0.9)	9(2.0)
16x24	281(2.1)	26(4.6)	231(2.0)	36(4.5)	287(2.1)	20(4.5)	258(2.0)	21(4.7)
32x24	277(4.4)	50(9.7)	520(5.5)	71(9.9)	276(4.5)	45(9.8)	568(5.7)	50(9.9)
64x24	>3000	586(39)	-	209(26.3)*5	>3000	14(19.8)	-	51(20.8)
128x24	-	-	-	727(90)*6	-	17(40)	-	120(48)

*1: with $\epsilon = 1e-10$, 229(8.8 sec) , *2: with $\epsilon = 1e-10$, 171(14.3 sec)
*3: with $\epsilon = 1e-9$, 405(39 sec) , *4: with $\epsilon = 1e-10$, 261(18.7 sec)
*5: with $\epsilon = 1e-10$, 137(23.0 sec), *6: with $\epsilon = 1e-10$, 313(59.0 sec)

TABLE 14. (I)-Solution of Stokes problem in a stretched backward facing step with Bicgstab

$Q2-Q1$	Sloan renumbering, $eps = 1e-4$							
Grid	p-last				p-last-level			
	ILU(0)	ILU(2)	Lumped	ILU(2),Lumped	ILU(0)	ILU(2)	Lumped	ILU(2),Lumped
8x24	22(0.7)	11(1.2)	22(0.8)	12(1.2)	13(0.8)	5(1.04)	14(0.8)	8(1.1)
16x24	44(1.7)	19(2.6)	38(1.6)	24(2.5)	30(1.7)	13(2.4)	29(1.6)	15(2.5)
32x24	538(5.5)	82(5.8)	115(3.6)	67(5.6)	226(4.1)	38(5.7)	156(3.8)	38(5.0)
64x24	-	150(13)	292(9.3)	145(12.5)	40(6.6)	15(9.5)	72(7.3)	44(9.9)
128x24	-	-	1420(41)	407(35)	265(18)	-	241(17.3)	91(21)
	Cuthill McKee renumbering, $eps = 1e-4$							
8x24	25(0.7)	11(1.7)	25(0.7)	13(1.6)	12(0.7)	7(1.6)	15(0.7)	8(1.6)
16x24	50(1.6)	19(3.7)	58(1.6)	26(3.6)	39(1.6)	9(3.3)	39(1.6)	12(3.4)
32x24	113(3.5)	36(7.6)	190(3.8)	46(7.7)	92(3.5)	18(7.2)	102(3.6)	30(7.2)
64x24	321(9.2)	142(19)	448(10.3)	140(19)	135(7.5)	-	609(11.9)	254(20)
128x24	1116(33)	-	-	398(51)	>3000	-	-	-

TABLE 15. (II)-Solution of Stokes problem in a stretched backward facing step with Bicgstab

Q2-Q1	BICGSTAB			GMRES		
Re=100	PCD	p-last-level(Sloan)	LSC	PCD	p-last-level(Sloan)	LSC
-	IFISS	SEPRAN	IFISS	IFISS	SEPRAN	IFISS
-	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)	Iter./t (sec)
8x24	32-0.9	10-0.1	12-0.5	32-0.5	16-0.03	15-0.33
16x24	44-3	14-0.11	10-1.1	33-1.21	21-0.08	15-0.76
32x24	56-8.8	39-0.67	12-3.1	37-3.16	68-0.67	18-2.1
64x24	58-21	33-1.21	16-7.7	45 - 8.3	61-1.13	25-9.1
Re = 200						
8x24	64-1.83	28-0.09	20-0.81	45-7.26	60-0.1	23-0.5
16x24	80-5.6	23-0.17	20-1.98	50-1.9	37-0.15	22-1.14
32x24	110-17.5	42-0.7	20-4.5	52-4.3	83-0.75	24-2.73
64x24	126-45	31-1.1	26-12.5	60-11	41-0.8	29-7

Table 16. Comparison of preconditioners used in IFISS and SEPRAN implemented in MATLAB

6. Conclusions

In this report, an approximate solution of the Stokes and Navier Stokes equations with iterative solvers is carried out using the ILU preconditioner. From the results it is clear that renumbering of mesh points and unknowns prevent the break down of the ILU preconditioners and lead to faster convergence with Krylov subspace methods. In the Stokes problem, the number of iterations increases with the increase number of elements in the grid, In the case of Navier Stokes equations, the convergence of preconditioned GMRESR and BiCGSTAB is better than preconditioned GMRES(m), especially when using high Reynolds numbers and large mesh size. For non-linear iterations, it is better to use a few Picard's iterations followed by Newton's iterations. It is a good idea to use an iterative solver with low accuracy for the linearized problem, because of a high accuracy has no effect on the convergence of the outer iterations. A direct solver works efficiently in 2-D for small problems, but it is not good idea to solve large 2-D or 3-D problems at the cost of large memory and time requirement. Note that our preconditioner has been used in IFISS for stabilized finite element and the results are comparable with results mentioned in this report. Vary stretched grids may have a negative influence on the convergence behavior. This subject requires future research.

References

[1] http://ta.twi.tudelft.nl/sepran/sepran.html

[2] G. Segal. *SEPRAN Introduction*. Leidschendam, NL: Ingenieursbureau Sepra; 1995.

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

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

[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] 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.

[7] W. E. Arnoldi. The principle of minimized iteration in the solution of the matrix eigen problem. *Quart. Appl. Math. , 9*,17-29,1951.

[8] G. H. Golub, C. F. Van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, 1996.

[9] G. Meurant. *Computer solution of large linear systems*. In Studies in Mathematics and its Applications, Vol. 28, Lions JL , Papanicolaou G , Fujita H , Keller HB (eds). Elsevier: Amsterdam, 1999.

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

[11] H. A. Van Der Vorst, C. Vuik. GMRESR: a family of nested GMRES methods. *Num. lin. Alg. Appl. , 1*,369-386,1994.

[12] R. Fletcher. Factorizing symmetric indefinite matrices. *Lin. alg. and its Appl. , 14*,257-277,1976.

[13] P. Sonneveld. CGS: a fast Lanczos-type solver for nonsymmetric linear systems. *SIAM J. Sci. Stat. Comput. 10*,36-52,1989.

[14] 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.

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

[16] 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.

[17] E. Cuthill, J. McKee. Reducing the bandwidth of sparse symmetric matrices. *Proc. ACM Nat. Conf. Association of Computing Machinery, New York*, 1969.

[18] G. Segal,C. Vuik. A simple iterative linear solver for the 3D incompressible Navier Stokes equations discretized by the finite element method. *Technical Report TUDelft*; 95-28.

[19] C. C. Paige, M. A. Saunders. Solution of sparse indefinite system of linear equations. *SIAM J. Num. Anal. 12*,617-629,1975.

[20] D. Kay, D. Loghin, A. Wathen. A preconditioner for the steady state Navier Sotkes equations. *SIAM J. Sci. Comput. 24*, 237-256, 2002.

[21] D. Silvester, H. Elman, D. Kay, A. Wathen. Efficient preconditioning of the linearized Navier-Stokes for the incompressible flow. *J. Comp. Appl. Math. 128*, 261-279, 2001.

[22] H. C. Elman. Preconditioning for the steady-state Navier Stokes equations with low viscosity. *SIAM J. Sci. Comput. 20,* 1299-1316, 1999.

[23] H. C. Elman, V. E. Howle, J. Shadid, D. Silvester, R. Tuminaro. In preparation, 2006. *Block preconditioner based on approximate Commutators*. Tech. rep., Institute for Advanced Studies, University of Maryland.

[24] G. L. G. Sleijpen, D. R. Fokkema. BICGSTAB(ℓ) for linear equations involving unsymmetric matrices with complex spectrum.*Elec. Numer. Math. 1*, 11-32, 1993.

[25] T. Kelley. *Iterative Methods for linear and nonlinear equations*, SIAM, Philadelphia, 1995.

Appendix A. Mesh Numbering Schemes

Before going into detail of algorithm, it is appropriate to state some basic definitions. A *graph* G is defined to be pair $(N(G), E(G))$ where $N(G)$ is a non-empty finite set of members called *nodes*, and $E(G)$ is a finite set of unordered pairs, comprised of distinct members of $N(G)$, called *edges*. A graph comprises of unordered pairs is called *undirected graph*. The *degree* of node i in G is defined as the number of edges incident to i. A path in G is defined by a sequence of edges such that consecutive edges share a common node. A graph G is connected if each pair of distinct nodes is connected. The *distance* between i and j is denoted $d(i, j)$, and is defined as the number of edges on the shortest path connecting them. The *diameter* of G is defined as the maximum distance between any pair of nodes. Nodes which are at the opposite ends of the diameter are called *peripheral* nodes. A *rooted level structure* is defined as the partitioning of $N(G)$ into levels $l_1(r), l_2(r), ..., l_h(r)$ such that:

- $l_1(r) = r$ where r is the root node of the level structure
- for $i > 1$, $l_i(r)$ is the set of all the nodes not yet assigned level, which are adjacent to nodes in $l_{i-1}(r)$.

The *depth* h is defined as total number of levels. The width of a level is defined as total number of nodes in one level. A width of the level structure is defined as maximum number of nodes in level structure.

$$w = \max_{1 \le i \le h} |l_i(r)|$$

A.1. **Sloan's Algorithm.**

A.1.1. *STEP 1 (Selection of pseudo diameter).*

(1) Choose a node s with minimum degree.
(2) build a level structure $\mathcal{L}(s) = L_0(s), ..., L_k(s)$,
(3) sort the nodes of $L_k(s)$ by increasing degree, let m be the number of the elements in $L_k(s)$ and Q be the $[\frac{m+2}{2}]$ first elements of the sorted set,
(4) Let $w_{min} = \inf$ and $k_{max} = k$. For each node $i \in Q$ in order of ascending degree, generate $\mathcal{L}(s) = L_0(s), ..., L_k(s)$. if $k > k_{max}$ and $w = \max_{i \le j \le k} |L_j(i)| < w_{min}$, then we set $s = i$ and go to step 3. Otherwise, if $w < w_{min}$, we set $e = i$ and $w_{min} = w$

We exit this algorithm with a starting node s and an end node e which define a pseudo diameter.

A.1.2. *STEP 2(node labeling).* The nodes are classified in four catogories according to their status. Node which are been already assigned label are *postactive*. Nodes which have not been assigned a number but are adjacent to the postactive nodes are *active* . Nodes without a number adjacent to active nodes are *preactive* . All other nodes are *inactive* . The current degree n_i of a node i is defined as
$n_i = m_i - c_i + k_i$,
where m_i is the degree of i, c_i is the number of the postactive or active nodes adjacent to i and $k_i = 0$ if i is active or postactive and $k_i = 1$ otherwise. The input to the following algorithm are the two nodes s and e selected in STEP 1.

(1) for all nodes, compute the distance $d(e, i)$ from i to e, initialize all nodes as inactive and set

$$P_i = (n_{max} - n_i) * W_1 + d(e, i) * W_2,$$

where $n_{max} = \max_{1 \le i \le N} n_i$. For convenience n_{max} may be set equal to N(since the maximum current degree in any graph with N nodes is N. W_1

and W_2 are integer weights. The queue of eligible nodes is initialized with s which is assigned a preactive status.

(2) as long as the queue is not empty,
   2.1 select the node i with highest priority(nodes with low current degree and large distance to the end have high priorties, ties are broken arbitrarily),
   2.2 delete i from the queue. If it is not preactive, go to 2.3. Else, consider each node j adjacent to i and set $P_j = P_j + W_1$. If j is inactive, insert j in the queue and declare it preactive,
   2.3 label node i and declare it postactive,
   2.4 Examine every node j adjacent to i. If j is preactive, set $P_j = P_j + W_1$, declare j as active and examine each node k adjacent to j. If k is active or preactive, set $P_k = P_k + W_1$, otherwise if k is inactive, set $P_k = P_k + W_1$, insert k in the queue and declare it as preactive.

recommended values of W_1 and W_2 are 2 and 1, respectively.

A.2. **Cuthill and McKee's Algorithm.** The Cuthill Mckee(CMK) algorithm is a local minimization algorithm whose aim is to reduce the profile of matrix. The starting level can be a node or number of nodes which constitutes boundary of certain region.
ALGORITHM:

(1) Choose the starting node.
(2) for $i = 1, \ldots.., n-1$ number all the non-numbered neighbors of x_i in increasing order of degree.
(3) Update the degrees of the remaining nodes

Appendix B. SEPRAN Introduction

SEPRAN is developed at "Ingenieursbureau SEPRA" and Delft University of Technology [2]. It is a general purpose finite element package. It enables the simulation of a wide variety of problems in fluid mechanics, structural mechanics and electromechanics. Two-dimensional, axi- symmetric and three-dimensional steady state or transient simulations in complex geometries can be carried out without any problem. The size of the problems is only limited by the computer time available and the capacity of the secondary storage devices. SEPRAN provides a wide range of possible analysis, for example:

- Potential problems (potential flow, ground water flow, electromagnetism)
- Second order elliptic equations (diffusion, convection-diffusion) both linear and nonlinear.
- Isothermal or nonisothermal Newtonian or non-Newtonian incompressible flow governed by the Navier-Stokes equations
- Free boundary and moving boundary problems
- Helmholtz-type equations
- structural analysis
- Lubrication analysis (compressible, incompressible) - Solidification problems
- Heat equations (i.e. time-dependent nonlinear second order equations)
- Stability analysis