Fast Iterative Methods

for

The Incompressible Navier-Stokes Equations

Proefschrift

ter verkrijging van de graad van doctor
aan de Technische Universiteit Delft,
op gezag van de Rector Magnificus prof.ir. K.C.A.M. Luyben,
voorzitter van het College van Promoties,
in het openbaar te verdedigen op woensdag 24 februari 2010 om 12.30 uur

door

Mehfooz ur REHMAN,
Master of Science (M.Sc.) Systems Engineering, Pakistan Institute of Engineering
and Applied Sciences, Quaid-i-Azam University Islamabad, Pakistan

geboren te Kohat, Pakistan.

Dit proefschrift is goedgekeurd door de promotor:
Prof.dr.ir. C. Vuik

Copromotor:
Ir. A. Segal

Samenstelling promotiecommissie:

Rector Magnificus	voorzitter
Prof.dr.ir. C. Vuik	Technische Universiteit Delft, promotor
Ir. A. Segal	Technische Universiteit Delft, copromotor
Prof.dr.ir. C. W. Oosterlee	Technische Universiteit Delft
Prof.dr. W. H. A. Schilders	Technische Universiteit Eindhoven
Prof.dr. A. E. P Veldman	Rijksuniversiteit Groningen
Dr. A. P. van den Berg	Universiteit Utrecht
Prof.dr.ir. S. Vandewalle	Katholieke Universiteit Leuven, België
Prof.dr.ir. C. R. Kleijn	Technische Universiteit Delft, reservelid

This thesis has been completed in partial fulfillment of the requirements of Delft University of Technology (Delft, The Netherlands) for the award of the Ph.D. degree. The research described in this thesis was supported by Delft University of Technology, and Higher Education Commission (HEC) Pakistan. I thank them sincerely for their support.

Fast Iterative Methods for The Incompressible Navier-Stokes Equations.
Dissertation at Delft University of Technology.

ISBN # 978-90-9024925-4

Cover: A numerical solution of 2D driven cavity flow Stokes problem on 132×132 Q2-Q1 FEM grid.

Summary

Efficient numerical solution of the incompressible Navier-Stokes equations is a hot topic of research in the scientific computing community. In this thesis efficient linear solvers for these equations are developed.

The finite element discretization of the incompressible Navier-Stokes equations gives rise to a nonlinear system. This system is linearized with Picard or Newton type methods. Due to the incompressibility equation the resulting linear equations are of saddle point type. Saddle point problems also occur in many other engineering fields. They pose extra problems for the solvers and therefore efficient solution of such systems of equations forms an important research activity. In this thesis we discuss preconditioned Krylov methods, that are developed for saddle point problems.

The most direct and easy applicable strategy to solve linear system of equations arising from Navier-Stokes is to apply preconditioners of ILU-type. This type of preconditioners is based on the coefficients of the matrix but not on knowledge of the system. In general, without precautions, they fail for saddle point problems. To overcome this problem, pivoting or renumbering of nodal points is necessary. Direct methods also suffer from the same problem, i.e zeros may arise at the main diagonal. Renumbering is used to reduce the profile or bandwidth of the matrix. To avoid zero pivots it is necessary to use extra information of the discretized equations. First we start with a suitable node renumbering scheme like Sloan or Cuthill-McKee to get an optimal profile. Thereafter unknowns are reordered per level such that zero pivots move to the end of each level. In this way unknowns are intermixed and the matrix can be considered as a sequence of smaller subsystems. This provides a reasonable efficient preconditioner if combined with ILU. We call it Saddle point ILU (SILU).

A completely different strategy is based on segregation of velocity and pressure. This is done by so-called block preconditioners. These preconditioners are all based on SIMPLE or Uzawa type schemes. The idea is to solve the coupled system with a Krylov method and to accelerate the convergence by the block preconditioners. The expensive steps in the preconditioning is the solution of the velocity and pressure subsystem. The subsystems may be solved by direct methods, Krylov methods or multigrid. We employ SIMPLE-type preconditioners that are based on the classical

SIMPLE method of Patankar. Convergence with the SIMPLE method depends on relaxation parameters that can only be chosen by trial and error. Since our preconditioner is based on only one step of a SIMPLE iteration, we predict that there is no need for a relaxation parameter. We suggest several improvements of the SIMPLE preconditioner, one of them, MSIMPLER, appears to be very successful.

To test the preconditioners we use the classical benchmark problems of driven cavity flow and backward facing step both in 2D and 3D. We compare our preconditioners (SILU and MSIMPLER) with the popular LSC preconditioner, which is considered to be one of the most efficient preconditioners in the literature. SILU is combined with Bi-CGSTAB(ℓ) and IDR(s) a new method based on the Induced Dimension Reduction (IDR) algorithm proposed by Sonneveld in 1980. In cases where Bi-CGSTAB(ℓ) shows poor convergence, IDR(s) in general behaves much better.

Physical problems with slowly flowing materials, like for example mantle convection in the earth, may be modeled with the variable viscosity Stokes equations. In this case specially adapted preconditioners are required. In this thesis we present some new preconditioners all based on the pressure mass matrix approximation of the Schur complement matrix. Special emphasis is required for scaling and stopping criteria in combination with variable viscosity. The new methods are tested on various classes of problems with different viscosity behavior. They appear to be independent of the grid size and the viscosity variation.

Samenvatting

Het efficiënt numeriek oplossen van de incompressibele Navier-Stokes vergelijkingen is een hot topic research onderwerp in de scientific computing gemeenschap. In dit proefschrift ontwikkelen we efficiënte lineaire solvers voor deze vergelijkingen.

De eindige elementen discretisatie van de incompressibele Navier-Stokes vergelijkingen resulteert in een niet-lineair systeem. Dit systeem wordt gelineariseerd met Picard of Newton methodes. Vanwege de incompressibiliteitsconditie zijn de resulterende lineaire vergelijkingen van het zadelpunt type. Zadelpuntsproblemen treden ook op in veel andere technische vraagstukken. Zij veroorzaken extra problemen in de solvers en daarom vormt het efficiënt oplossen van zulke vergelijkingen een belangrijke research activiteit. In dit proefschrift bediscussiëren wij gepreconditioneerde Krylov methoden welke speciaal voor zadelpuntsproblemen zijn ontwikkeld.

De meest directe en eenvoudigste strategie om lineaire stelsels vergelijkingen, welke ontstaan door discretisatie van Navier-Stokes, op te lossen is om preconditioners van het ILU-type toe te passen. Dit type preconditioners is gebaseerd op de coëfficiënten van de matrix, zonder kennis van het onderliggende probleem. Zonder bijzondere voorzorgsmaatregelen falen zij in het geval van zadelpuntsproblemen. Teneinde dit te voorkomen, is het noodzakelijk om te pivoteren, dan wel knooppunten te hernummeren.

Ook directe methodes hebben last van hetzelfde euvel, namelijk er komen nullen voor op de hoofddiagonaal. Hernummeren van knooppunten wordt toegepast om het profiel of de bandbreedte van de matrix te reduceren. Als we willen voorkomen dat pivots nul worden, is het nodig extra informatie van de gediscretiseerde vergelijkingen te gebruiken. Teneinde een optimaal profiel te krijgen, starten we met een geschikt hernummeringsalgorithme zoals Sloan of Cuthill-McKee. Daarna worden de onbekenden per level herordend, zodat pivots die nul zijn naar het einde van ieder level worden verplaatst. Op deze manier worden de onbekenden verwisseld en kan de matrix opgevat worden als een stelsel van kleinere subsystemen. In combinatie met ILU ontstaat een redelijk efficiënte preconditioner, die wij Saddle point ILU (SILU) noemen.

Een geheel andere strategie is gebaseerd op de scheiding van snelheid en druk onbekenden. Dit wordt gedaan met behulp van zogenaamde blokpreconditioners.

Al deze preconditioners zijn gebaseerd op SIMPLE dan wel Uzawa type schema's. Het idee is om het gekoppelde systeem op te lossen met een Krylov methode en de convergentie te versnellen met behulp van de blokpreconditioners. Het oplossen van de substelsels van de snelheid en druk vormt het rekenintensieve deel van de preconditionering. De substelsels kunnen worden opgelost met behulp van directe methodes, Krylov methodes of multirooster. Wij passen SIMPLE-achtige preconditioners toe, gebaseerd op de klassieke SIMPLE methode van Patankar. De convergentie van SIMPLE hangt af van relaxatieparameters die door trial-and-error gekozen moeten worden. Omdat onze preconditioner is gebaseerd op slechts één stap van een SIMPLE iteratie, is relaxatie niet nodig. We suggeren verscheidene verbeteringen van de SIMPLE preconditioner, waarvan één, MSIMPLER, erg successvol blijkt te zijn.

Om de preconditioners te testen gebruiken we twee klassieke benchmark problemen, te weten het driven cavity problem en de backward facing step, zowel in 2D als 3D. We vergelijken onze preconditioners (SILU en MSIMPLER) met de populaire LSC preconditioner, welke in de literatuur als een van de meest efficiënte preconditioners wordt aangemerkt. SILU wordt gecombineerd met Bi-CGSTAB(ℓ) en ook met IDR(s), een nieuw algorithme gebaseerd op het Induced Dimension Reduction (IDR) algorithme van Sonneveld (1980). In die gevallen waar Bi-CGSTAB(ℓ) slecht convergeert, blijkt IDR(s) in het algemeen veel beter te presteren.

Fysische problemen met langzaam stromende materialen, zoals bijvoorbeeld mantel convectie in het aardoppervlak, kunnen gemodelleerd worden met de Stokes vergelijkingen met variabele viscositeit. In dat geval zijn speciale preconditioners vereist. In dit proefschrift presenteren we enkele nieuwe preconditioners, alle gebaseerd op de approximatie van de Schur complement matrix door de massamatrix van de druk. Voor variabele viscositeits problemen is het noodzakelijk speciale aandacht te besteden aan schaling en afbreekcriteria. De nieuwe methodes worden getest op verschillende probleemklassen met hun specifiek viscositeitsgedrag. Zij blijken onafhankelijk te zijn van roosterafmeting en viscositeitsvariatie.

ACKNOWLEDGMENTS

I would like to thank my supervisor, Prof. Dr. Ir. Kees Vuik and co-supervisor, Ir. Guus Segal for their supervision and support during my PhD. Their professional help provided a jump start to my PhD-research. It all started by their introducing some nice pointers to the literature. I would specifically like to mention the IFISS package, Benzi's work on saddle point problems that was published in 2005 (start year of this program), Kees Vuik's work on SIMPLE-type preconditioners, and the SEPRAN package. I learnt a lot of new things from them, and regular professional meetings with them helped me in identifying areas having research potential. I found them ever-welcoming in helping me solve my problems, both technical and social. I am very grateful to them for their help, and would therefore like to take this opportunity to express it formally.

As I consider my doctoral thesis a big achievement in my life, I would also like to mention some people who had a share in making it happen. First of all, many thanks, sincere prayers, and a lot of love and gratitude to my parents for their gross untiring efforts to educate us. Their kindness and sympathy can neither be expressed, nor can be amply thanked in words. They took good care of my family when I was estranged from them at the start of my PhD, until the time when their re-union with me became possible. I would also like to thank all my close and distant family who prayed for my success and missed me on special occasions. Many thanks to my wife's family for their equal support in making us comfortable here. My late uncle Ahmed Jan, had a jolly personality through which he taught me how to see the lighter side of life in days of gloom. Many prayers for him, his lessons about life would stay kindled in my heart and mind always.

In the Netherlands, after a hard first year, my house turned into a home by the arrival of my wife, daughter and son. I thank my wife for managing all the activities that would have hindered my progress. I revelled in their company and I am sure we will remember this period of our life very much. We all spent a cherishable time together in the Netherlands.

There was a lot of social support and community life through the acquaintance of many Pakistani friends and families that I came to know during my Dutch stay. I am

happy that I found many friends. I thank them all for sharing very nice times with me and my family. We enjoyed parties including Iftar and Eid gatherings. My weekends were usually engaged by some of my cricketer friends (both Indians and Pakistanis). Thanks to all of them. I also thank my friends in Germany with whom I shared very fruitful time during my visits to Germany.

In the department, I would like to thank Coen Leentvaar, my ex-office mate, who helped me in a lot of diverse issues, specially in translating Dutch letters that I received from time to time from various organizations. I also thank Hisham bin Zubair for helping me out in many matters during my PhD. Besides, I feel privileged to be a part of the numerical analysis research group at Delft with the presence of Piet Wesseling, Kees Oosterlee and P. Sonneveld. I wish to thank all group members with whom I shared office, participated in conferences, enjoyed birthday parties and coffee breaks. I thank Diana and Mechteld for providing assistance in numerous ways. Thanks to Kees Lemmens for providing us with an error free network.

I would also like to thank Thomas Geenen from Utrecht and Scott MacLachlan for their fruitful discussion on preparation of the Stokes papers. We, (I and Thomas), shared many ideas and exchanged them electronically. This assisted us a lot us in understanding issues related to iterative solvers.

Many thanks to Franca Post from CICAT and Loes Minkman from NUFFIC for dealing our scholarship matters in a very efficient way.

I would like to thank the committee members for sparing time to read the manuscript. I thank Kees Oosterlee for his helpful comments, which led to considerable improvements in the manuscript. Besides, Guus Segal's help in providing me Dutch translation *Samenvatting* and *Stellingen* is highly appreciated.

All this would have not been possible without the help and willingness of Almighty Allah. I thank Allah for His blessings on me.

Mehfooz ur Rehman
Delft, September 21, 2009

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The Navier-Stokes equations form the basis for modeling both laminar as well as turbulent flows. Depending on the Reynolds number a flow is characterized as either laminar or turbulent. A fluid such as air, water or blood is called incompressible if a large force acting on this fluid fails to impact a change in its volume. In general, a fluid is considered to be (nearly) incompressible if the speed of the fluid is small (≤ 0.1) compared to the speed of sound in that fluid. The Navier-Stokes equations are used to simulate various physical phenomena, for example, weather prediction, geodynamic flows, and aneurysm in blood vessels. Although the scope of this thesis is limited to laminar, and incompressible flows, the techniques that we develop and apply here may also be used for turbulent flows. Except for some simple cases, analytical solution of the Navier-Stokes equations is impossible. Therefore, in order to solve these equations, it is necessary to apply numerical techniques. To that end, numerical discretization methods like Finite Difference Methods (FDM), Finite Volume Methods (FVM) and Finite Element Methods (FEM) are usually applied as standard practice. For relatively simpler problems analytical solution can be used to verify the numerical results. In this thesis we shall focus ourselves on discretization by the FEM.

The discretization of the Navier-Stokes equations leads to a nonlinear system of equations. The solution process therefore involves the linearization of such a nonlinear system, which is followed by an efficient solution of the resulting matrix equation $Ax = b$.

Direct solution methods give the exact numerical solution of this system $x = A^{-1}b$. Although each distinct direct method has a different route of reaching this, they all have a common denominator in terms of memory and CPU time expense. Obtaining the solution of large problems with direct solvers is therefore not viable. The alternative is to apply iterative techniques that approximate the solution to the desired accuracy. For large systems, iterative methods are usually cheap but less robust compared to direct methods. There are three major classes of iterative methods, classical stationary iterative methods such as Jacobi, Gauss Seidel, SOR etc., non-stationary

iterative methods which includes the family of Krylov subspace methods, and multilevel iterative correction techniques which include multigrid and AMG. An efficient scheme may consist of one of the methods from these classes, but also of a combination of solvers, for example, multigrid preconditioned Krylov methods are often used as a solver of choice in many versatile situations. A survey of such methods can be found in [67, 83, 5, 64].

1.1 Open problem

The advancement in computer hardware (high speed processors, large memory etc.) has enabled researchers to obtain numerical solutions of many complex problems. This achievement has contributed a lot to the third way of fluid dynamics (Computational Fluid Dynamics known as CFD) which was previously based mostly on experimental and theoretical setups. Patankar [60] was in 1980 one of the pioneers in developing algorithms for a fast solution of the incompressible Navier-Stokes equations. Compared to experiments, CFD has reduced the cost and improved simulation techniques and therefore provides better insight of the problem. Problems that would have taken years to understand in experimental setups are now simulated in days.

One of the challenges of the last few decades is the construction of fast numerical solution algorithms for the incompressible Navier-Stokes equations. The discretization and linearization of Navier-Stokes gives rise to a saddle point problem with a zero block on the main diagonal due to the absence of the pressure in the continuity equation. Saddle point problems also arise in electrical circuits, linear elasticity, constraint optimization and many other fields. A survey on saddle point problems is given by Benzi [9]. Due to its specific character and its appearance in many engineering fields, solution of saddle point problems is a prominent subject in the numerical research field. In case of the Navier-Stokes problem, SIMPLE-type and Uzawa-type methods are well-known in the literature [60], [4]. These methods decouple the system and solve the subsystem for the velocity and pressure separately.

Recent developments in the Krylov method and multigrid has improved the efficiency of iterative methods. Coupled systems are solved with the help of a preconditioned Krylov method or efficient multigrid techniques. In the Navier-Stokes problem, the final goal is to develop solvers that converge independently of mesh size and Reynolds number.

In terms of preconditioning strategies, the most common and easy strategy is to apply an algebraic preconditioner to the coupled system. Usually such preconditioners rely on information present in the coefficient matrix without having complete knowledge of the system. Such preconditioners can be easily adapted for a variety of problems. Incomplete LU (Gaussian elimination) variants and approximate inverse (AINV) are the good examples of such preconditioners [52], [11]. Convergence with such preconditioners can be made efficient by using renumbering of the grid points or applying pivoting techniques. In general such renumbering scheme are developed for direct solvers to reduce the profile and bandwidth of the matrix [53, 29, 94]. However

these schemes have also been efficiently used to enhance the convergence of the ILU preconditioned Krylov method [51, 11, 14, 25, 93]. In this thesis we develop efficient renumbering schemes for the incompressible Navier-Stokes problem.

Another popular strategy, known as block preconditioner, is based on a segregation approach. SIMPLE and Uzawa-type schemes are the basis for such preconditioners. A coupled system is solved with the help of a Krylov method that is accelerated with the help of block preconditioners. The expensive component of these block preconditioners is the solution of the velocity and pressure subsystems. The pressure subsystem arises due to an appropriate Schur complement approximation. The subsystems may be solved directly, or through an iterative approach, such as by using a Krylov method or a multigrid technique. Older schemes like SIMPLE and Uzawa are been used as part of iterative methods by performing efficient precondioning steps [90], [35], [10]. These methods are also used as smoothers in some multigrid techniques [92, p. 298], [15], [42]. In our work, we focus on improving convergence with block preconditioners. SIMPLE and block triangular preconditioners are employed. A block triangular preconditioner is a special form of an Uzawa method, in which first the pressure subsystem is solved and then the velocity subsystem is solved after updating the right-hand side with the pressure obtained from the first step. The main part of this type of preconditioners is the efficient solution of the subsystem corresponding to velocity and pressure. Multigrid (MG) or preconditioned Krylov methods can be employed to solve such systems.

Besides Navier-Stokes with constant viscosity, the variable viscosity Stokes problem models physical processes in geodynamics, for example mantle convection in the earth. In geodynamical processes, viscosity varies due to change in material properties at short distances. The sharp varying viscosity makes the problem challenging for the scientific computing community. Much research is going on to solve such problems [50], [19], [57]. We apply our schemes to problems with different viscosity configurations, including an extrusion problem that has a relatively smooth varying viscosity.

1.2 Outline of the thesis

The thesis is divided in the following chapters.

- In Chapter 2, the model equation, finite element dicretization and linearization of the incompressible Navier-Stokes equations are discussed.

- Linear solvers (direct, classical, Krylov, multigrid) and preconditioner introduction form the subject of Chapter 3.

- Since we are interested in preconditioners for the incompressible Navier-Stokes problem, in Chapter 4 we give a brief overview of some important preconditioners both algebraic and physics-based.

- We discuss the saddle point ILU (SILU) preconditioner in Chapter 5. This is a cheap and easy to implement ILU preconditioner in combination with a well chosen renumbering strategy.
- Chapter 6 deals with block preconditioners that are based on SIMPLE-type formulations. Important improvements in SIMPLE-type preconditioners are discussed.
- A comparison of the preconditioners for Navier-Stokes in 2D and 3D is done in Chapter 7. Preconditioners are also tested for stretched grids. Comparison of SILU preconditioned IDR(s) and Bi-CGSTAB(ℓ) is also part of this chapter.
- Some promising techniques for the solution of the Stokes problem are discussed in Chapter 8. Preconditioners are applied to solve different problems with various viscosity configurations.
- Chapter 9 is devoted to conclusions.

Chapter 2

Finite element discretization and linearization of the Navier-Stokes equations

In this chapter, we formulate the steady state, incompressible Navier-Stokes equations. We shortly describe the discretization by finite element methods. It will be shown that the structure of the matrix depends on the selection of the elements. Newton and Picard techniques are used to linearize the system of nonlinear equations.

2.1 Problem description

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

$$-\nu\nabla^2\mathbf{u} + \mathbf{u}.\nabla\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in} \quad \Omega \tag{2.1}$$

$$\nabla.\mathbf{u} = 0 \quad \text{in} \quad \Omega. \tag{2.2}$$

$\Omega \subset \mathbf{R}^d (d = 2 \; or \; 3)$ is the flow domain with piecewise smooth boundary $\partial\Omega$, $\mathbf{u}$ 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 operator.

Equation (2.1) represents conservation of momentum, while Equation (2.2) represents the incompressibility condition, or mass conservation. The boundary value problem that is considered is the system (2.1, 2.2) 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

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

The presence of the convective term $\mathbf{u}.\nabla\mathbf{u}$ in the momentum equation makes the Navier-Stokes system nonlinear. 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

$$-\nabla^2\mathbf{u} + \nabla p = \mathbf{f} \quad \text{in} \quad \Omega \tag{2.3}$$

$$\nabla.\mathbf{u} = 0 \quad \text{in} \quad \Omega, \tag{2.4}$$

with boundary condition

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

2.2 Discretization

The discretization of the Navier-Stokes equations is done by the finite element method. The weak formation of the Navier-Stokes equations is given by:

$$\nu\int_\Omega (\nabla^2\mathbf{u}).\mathbf{v} + \int_\Omega (\mathbf{u}.\nabla\mathbf{u}).\mathbf{v} - \int_\Omega (\nabla p).\mathbf{v} = 0, \tag{2.5}$$

$$\int_\Omega q(\nabla.\mathbf{u}) = 0, \tag{2.6}$$

where $\mathbf{v}$ and q are test functions in velocity and pressure space, respectively. After applying the Gauss divergence theorem and substitution of the boundary conditions, (2.5) and (2.6) reduce to:

Find $\mathbf{u} \in H^1_E(\Omega)$ and $p \in L_2(\Omega)$ such that

$$\nu\int_\Omega \nabla\mathbf{u} : \nabla\mathbf{v} + \int_\Omega (\mathbf{u}.\nabla\mathbf{u}).\mathbf{v} - \int_\Omega p(\nabla.\mathbf{v}) = \int_{\partial\Omega_N} \mathbf{s}.\mathbf{v}, \tag{2.7}$$

$$\int_\Omega q(\nabla.\mathbf{u}) = 0. \tag{2.8}$$

H^1_E denotes the 2 or 3 dimensional Sobolev space of functions whose generalized derivatives are in $L_2(\Omega)$. The subscript E refers to the essential boundary condition. Subscript E_0 refers to homogeneous essential boundary conditions. : denotes the dyadic product. The discrete version of (2.7) and (2.8) is:

Given the finite dimensional subspaces $\mathbf{X}^h_0 \subset \mathbf{H}^1_{E_0}$, $\mathbf{X}^h \subset \mathbf{H}^1_E$ and $M^h \subset L_2(\Omega)$, find $\mathbf{u_h} \in \mathbf{X}^h_E$ and $p_h \in M^h$ such that:

$$\nu\int_\Omega \nabla\mathbf{u_h} : \nabla\mathbf{v_h} + \int_\Omega (\mathbf{u_h}.\nabla\mathbf{u_h}).\mathbf{v_h} - \int_\Omega p_h(\nabla.\mathbf{v_h}) = \int_{\partial\Omega_N} \mathbf{s}.\mathbf{v_h} \text{ for all } \mathbf{v_h} \in \mathbf{X}^h_0, \tag{2.9}$$

$$\int_\Omega q_h(\nabla.\mathbf{u_h}) = 0 \text{ for all } q_h \in M^h. \tag{2.10}$$

We see in the relations (2.9) and (2.10) that no derivative of p_h and q_h are used. It is sufficient that p_h and q_h are integrable. For $\mathbf{u_h}$ and $\mathbf{v_h}$, the integral of the first derivative must exist. So we need the continuity of $\mathbf{u_h}$ and $\mathbf{v_h}$ and not of p_h and q_h in the weak formulation. This plays an important role in the element selection.
In the standard Galerkin method we define two types of basis functions, $\psi_i(x)$ for the pressure and $\phi_i(x)$ for the velocity. So the approximation for $\mathbf{u_h}$ and p_h is defined as

$$p_h = \sum_{j=1}^{n_p} p_j\psi_j(x),\ n_p \text{ is the number of pressure unknowns} \tag{2.11}$$

and

$$\mathbf{u_h} = \sum_{j=1}^{\frac{n_u}{2}} u_{1j}\phi_{j1}(x) + u_{2j}\phi_{j2}(x) = \sum_{j=1}^{n_u} u_j\phi_j(x), \tag{2.12}$$

where n_u is the number of velocity unknowns, u_j is defined by $u_j = u_{1j}$, $j = 1, ..\frac{n_u}{2}$, $u_{j+\frac{n_u}{2}} = u_{2j}$, $j = 1, ...\frac{n_u}{2}$ and ϕ_j in the same way. Substituting $\mathbf{v} = \phi_i(x)$, $q = \psi_i(x)$, we get the standard Galerkin formulation.

Find p_h and $\mathbf{u_h}$, such that

$$\nu\int_\Omega \nabla\mathbf{u_h} : \nabla\phi_i + \int_\Omega (\mathbf{u_h}.\nabla\mathbf{u_h}).\phi_i - \int_\Omega p_h(\nabla.\phi_i) = \int_{\partial\Omega_N} \mathbf{s}.\phi_i \quad for\ i = 1, ..n_u, \tag{2.13}$$

$$\int_\Omega \psi_i(\nabla.\mathbf{u_h}) = 0 \quad for\ i = 1, .., n_p. \tag{2.14}$$

Formally the system of equations can be written as

$$A_d u + N(u) + B^T p = f \tag{2.15}$$

$$Bu = g, \tag{2.16}$$

where u denotes the vector of unknowns u_{1i} and u_{2i}, and p denotes the vector of unknowns p_i. $A_d u$ is the discretization of the viscous term and $N(u)$ the discretization of the nonlinear convective term, Bu denotes the discretization of minus the divergence of u and $B^T p$ is the discretization of the gradient of p. The right-hand side vectors f and g contain all 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 easily, equations (2.15) and (2.16), have to be linearized and combined with some iteration process.

2.3 Linearization schemes

The Navier-Stokes equations are solved by solving a linearized problem at each nonlinear step. Linearization is commonly done by Picard and Newton iteration schemes, or variants of these methods.

2.3.1 Picard method

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 defined as

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

Starting with an 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

$$-\nu\Delta u^{k+1} + (u^{k}.\nabla)u^{k+1} + \nabla p^{k+1} = f \ \text{ in } \ \Omega, \tag{2.17}$$

$$\nabla.u^{k+1} = 0 \ \text{ in } \ \Omega, \tag{2.18}$$

$k = 1, 2, ...$ No initial pressure is required.

If we use $u^0 = 0$, the first iteration corresponds to the Stokes problem (2.3), (2.4).

2.3.2 Newton method

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

$$u^{k} = u^{k-1} + \delta u^{k-1}.$$

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

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

Neglecting the quadratic term in δu, the linearized form of (2.1), (2.2) becomes:

$$\nu\Delta u^{k} + u^{k}.\nabla u^{k-1} + u^{k-1}.\nabla u^{k} + \nabla p^{k} = f + u^{k-1}.\nabla u^{k-1}, \tag{2.19}$$

$$\nabla.u^{k} = 0. \tag{2.20}$$

Equations (2.19) and (2.20) are known as the Newton linearization of the Navier-Stokes equations and continuity equation. The Stokes equations can be used as an initial guess. Newton's method gives quadratic convergence. However, convergence with Newton largely depends upon the initial guess. For high Reynolds numbers, the method does not converge due to a bad initial guess. In such a case few Picard iterations could be used as a start. 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 a larger Reynolds number. This method is known as the continuation method.

After linearization the system can be written as

$$Fu + B^T p = f,$$

$$Bu = g,$$

where $F = A_d + N(u^k)$ is the linearized operator and u^k is the solution of the previous iteration. In general nonlinear iteration consists of the following steps.

Algorithm 2.1 Solve $A_d u + N(u) + B^T p = f$ and $Bu = g$

1. Initialize *tolerance*, u and p (usually u and p are obtained by solving the Stokes problem)
$$\begin{bmatrix} A_d & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}$$
2. Linearize $N(u)$ using u from the previous step using a Picard or Newton linearization scheme to create the matrix $F$and the right-hand side $\tilde{f}$
3. Solve
$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \delta u \\ \delta p \end{bmatrix} = - \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} + \begin{bmatrix} \tilde{f} \\ g \end{bmatrix}$$
4. Update
$$\begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} u \\ p \end{bmatrix} + \begin{bmatrix} \delta u \\ \delta p \end{bmatrix}$$
5. $$\text{If } \left\| \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} - \begin{bmatrix} f \\ g \end{bmatrix} \right\| \leq tolerance \left\| \begin{bmatrix} f \\ g \end{bmatrix} \right\| \text{ Then Converged}$$
Otherwise Goto 2

2.4 Element selection conditions

Now that the problem of the nonlinear term is dealt with, the linear system arising from Algorithm 2.1 can be written as

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix} \tag{2.21}$$

Equation (2.21) shows another problem in the system of equations: the presence of zeros in the main diagonal. The zero block reflects the absence of the pressure in the continuity equation. As a consequence the system of equations may be underdetermined for an arbitrary combination of pressure and velocity unknowns. Systems of the form (2.21) are known as saddle point problems. In (2.14) we see that the number

of equations for the 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 (2.21) becomes rank deficient, so we infer that the number of pressure unknowns should never exceed the number of velocity unknowns irrespective of the grid size. To meet this criterion in general, the pressure should be approximated by interpolation polynomials that are at least one degree less than the polynomials for the velocity. One can show [35] that for certain combinations of velocity and pressure approximations, the matrix in (2.21) is singular even though the pressure has a lower degree polynomial approximation. An exact condition that elements must satisfy is known as the Brezzi-Babuška condition (BB condition) [6, 17]. This condition states that, for BB^T in (2.21) to be invertible it is necessary that
kernel(B^T)= 0, where B^T is $n_u \times n_p$.
kernel(B^T)= 0 means that B^T has rank n_p, and is equivalent to requiring

$$\max_{\mathbf{v}}(B\mathbf{v}, p) = \max_{\mathbf{v}}(\mathbf{v}, B^T p) > 0, \forall p. \tag{2.22}$$

The above relation in the framework of the finite element method is

$$\max_{\mathbf{v}\in V_h} \frac{(\nabla.\mathbf{v_h}, q_h)}{\|\mathbf{v_h}\|_{V_h}\|q_h\|_{Q_h}} > 0. \tag{2.23}$$

The above condition (2.23) allows the family of matrices to degenerate towards a singular system as $h \to 0$. The strict Brezzi-Babuška condition ensures that BB^T does not degenerate towards zero as h decreases. The modified form of (2.23) is

$$\inf_{q\in Q_h} \sup_{\mathbf{v}\in V_h} \frac{(\nabla.\mathbf{v_h}, q_h)}{\|\mathbf{v_h}\|_{V_h}\|q_h\|_{Q_h}} \geq \gamma \geq 0. \tag{2.24}$$

In practice it is very difficult to verify whether the BB condition is satisfied or not. Fortin [37] has given a simple method to check the BB condition on a number of elements, which states that an element satisfies the BB condition, whenever, given a continuous differentiable vector field $\mathbf{u}$, one can explicitly build a discrete vector field $\tilde{\mathbf{u}}$ such that

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

The elements used in a finite element discretization of the Navier-Stokes equations are usually subdivided into two families, one having a continuous pressure (Taylor-Hood family) [77] and the Crouzeix–Raviart family [22] having a discontinuous pressure approximation. In Figures 2.1 and 2.2, some of the popular elements of these families in 2D 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 pressure which can be discontinuous on element boundaries. For output purposes, these discontinuous pressures are averaged in vertices for all the adjoining elements. For details see [24].

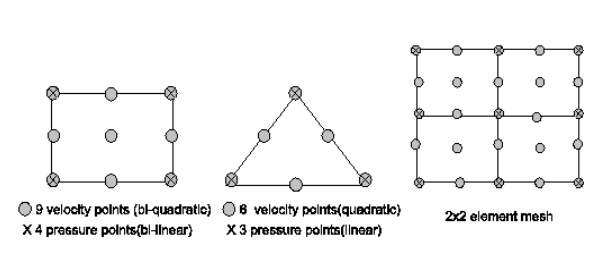


Figure 2.1: Taylor-Hood family elements (Q2-Q1) , (P2-P1) elements and (Q2-Q1) grid

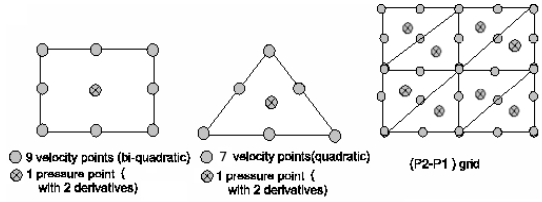


Figure 2.2: Crouzeix-Raviart family elements (Q2-P1), (P2-P1) elements and (P2-P1) grid

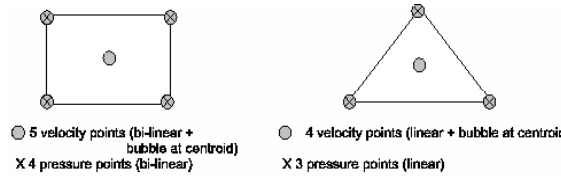


Figure 2.3: Taylor-Hood family mini-elements: $Q_1^+ - Q_1$ element, $P_1^+ - P_1$ element

Another class of elements from the Taylor-Hood family which satisfies the BB condition is known as the mini-element, in which the velocity is defined by a bilinear interpolation polynomial for the vertices with a bubble function at the centroid and the pressure is defined as a bilinear polynomial. The bubble function is 1 in the centroid and zero on the nodes and consequently, zero on the edges. This function 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 within which it exists. Therefore, the centroid can be eliminated on element level (static condensation). The rectangular and triangular mini-elements are shown in Figure 2.3. Elements that do not satisfy the BB condition must be stabilized in order to get a nonsingular system. Usually this is done by relaxing the incompressibility constraints. An example of such type of elements is the Q1-Q1 element in which equal order interpolation polynomials are used to approximate the velocity and pressure. The mini-element with static condensation is also an example of a stabilized element. Note that all elements used in this thesis satisfy the BB condition and stabilized elements are beyond the scope of the thesis.

2.5 Summary

In this chapter, we discussed the steady incompressible Navier-Stokes equations. Since we are interested in the numerical solution of the problem, we discretized it with a finite element discretization scheme. We use elements that satisfy the BB condition. Since discretization gives rise to a nonlinear system, Picard and Newton linearization schemes are used to linearize the Navier-Stokes problem. This gives rise to a saddle point problem which is indefinite and has a large number of zeros on the main diagonal.

In the next chapter, we will discuss solution techniques that can be employed to solve the linear systems.

Chapter 3

Solution techniques

Linearization of the Navier-Stokes problem, leads to a linear system of the form $\boldsymbol{A}x = b$ that needs to be solved in each step. In this chapter, we give an overview of various classes of solution methods that can be employed to solve a linear system.

3.1 Direct method

Each nonsingular matrix can be written in the form

$$\boldsymbol{A} = LU,$$

with L a lower triangular matrix and U an upper triangular matrix. Direct methods employ Gaussian elimination to construct L and U. After that we have to solve $LUx = b$, which can be done by first solving $Ly = b$, and then solving $Ux = y$.

The cost of the Gaussian elimination algorithm is $O(N^3)$ whereas $O(N^2)$ flops are used in backward and forward substitution, where N is the number of unknowns.

A direct method is preferred when the matrix is dense. However, sparse linear systems with suitable sparsity structure are also often solved by direct methods, since direct methods lead to a more accurate solution and a fixed amount of work compared to iterative methods. For sparse matrices, sparsity can be used to reduce the computing time and memory during the 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.

$$\boldsymbol{A} = \begin{bmatrix} x & x & 0 & 0 & 0 & 0 & 0 \\ x & x & x & 0 & 0 & 0 & 0 \\ x & x & x & x & 0 & 0 & 0 \\ 0 & x & x & x & x & 0 & 0 \\ 0 & 0 & x & x & x & x & 0 \\ 0 & 0 & 0 & x & x & x & x \\ 0 & 0 & 0 & 0 & x & x & x \end{bmatrix} \tag{3.1}$$

In (3.1), matrix $\boldsymbol{A}$ is a band matrix with 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 LU decomposition can now be obtained using $2Npq$ flops if $N \gg p$ and $N \gg q$. The solution of the lower triangular system costs $2Np$ flops and the upper triangular system costs $2Nq$ flops.

Linear systems arising from finite element and finite difference discretizations 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 the fact that L and U inherit the lower and upper bandwidth of $\boldsymbol{A}$. The cost of the banded solution methods is governed by the 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. This method is known as profile or envelope method.

For a square matrix $\boldsymbol{A}$, the lower envelope of $\boldsymbol{A}$ 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 $\boldsymbol{A}$ 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 elements above the main diagonal excluding leading zeros in each column. If a matrix is symmetric and positive definite then $\boldsymbol{A} = LL^T$, where L is the lower triangular matrix. This is known as the Cholesky factorization. In Cholesky factorization, L has the same envelope as $\boldsymbol{A}$ and we can save computer storage by employing a data structure that stores only the half band (lower or upper) of $\boldsymbol{A}$ and L can be stored over $\boldsymbol{A}$.

Generally, the system arising from the discretization contains a large number of zeros. Both band and profile storage depend on the order in which the equations and unknowns are numbered. The elimination process in the LU factorization fills the nonzero entries of a sparse matrix within a band or profile. So a large number of entries has to be stored and CPU time increases when the number of nodes increases. The aim of sparse direct solvers is to avoid doing operations on zero entries and therefore to try to minimize the number of fill-in. 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. Cuthill-McKee, Nested dissection, and some other renumbering schemes are widely used in the literature to reduce fill-in and cost of the direct solver. More details on direct solvers can be found in [27], [53], [40].

Although renumbering schemes may increase the efficiency of direct solvers considerably, in case of large problems, memory and CPU requirements still make their solution expensive. Especially in 3D, as well as in the case where a high accuracy is not required it is useless to apply direct methods.

3.2 Iterative methods

Suppose we want to solve a linear system

$$\boldsymbol{A}x = b. \tag{3.2}$$

We assume that $\boldsymbol{A}$ is a nonsingular square matrix and b is given. An iterative method constructs a sequence of vectors x_k, $k = 0, 1, ...$ (x_0 given), which is expected to converge towards x. The method is said to be convergent if

$$\lim_{k\to\infty} \|x - x_k\| = 0.$$

In many cases, the matrix A is split into two matrices

$$\boldsymbol{A} = \boldsymbol{M} - \boldsymbol{N}.$$

The sequence x_k can be defined as

$$\boldsymbol{M}x_{k+1} = \boldsymbol{N}x_k + b. \tag{3.3}$$

Let $e_k = x - x_k$ be the error at the kth iteration. Then (3.3) can be written as

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

$$e_{k+1} = \boldsymbol{M}^{-1}\boldsymbol{N}e_k$$

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

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

Theorem 3.2.1. *The iterative method (3.3) converges to $x = \boldsymbol{A}^{-1}b$ if $\sigma(\boldsymbol{M}^{-1}\boldsymbol{N}) < 1$ where $\sigma(\boldsymbol{M}^{-1}\boldsymbol{N}) = max\{|\lambda|,$ where λ is an element of the spectrum $\boldsymbol{M}^{-1}\boldsymbol{N}\}$; the set of eigenvalues of $\boldsymbol{M}^{-1}\boldsymbol{N}$ is said to be the spectrum of $\boldsymbol{M}^{-1}\boldsymbol{N}$ [87].*

It is not easy to inspect the spectrum, since for most of the problems the eigenvalues of $(\boldsymbol{M}^{-1}\boldsymbol{N})$ are not explicitly known. For more details see ([53], Chapter 5). Variants of (3.3) are known as classical iterative methods. Gauss Seidel, Gauss Jacobi and SOR (successive over relaxation) are examples of such classical methods. Advantages of iterative methods are:

- The matrix $\boldsymbol{A}$ is not modified, so no fill-in is generated and there is no need for additional space for new elements. Therefore, neither additional time nor memory is required for inserting these elements into a complicated data structure is required.
- Only a limited additional memory is required.
- In large problems or if only a low accuracy is required, these iterative methods may be much faster than direct methods.
- They are easy to implement.

Disadvantages of iterative methods are that convergence is not guaranteed for general matrices. Moreover, classical iterative methods may require a lot of time especially if a high accuracy is required. Classical iterative methods are used as smoothers in a multigrid method, where they are used to damp high spatial frequencies of the error.

In the next section, we consider a more sophisticated class of iterative solvers, known as Krylov subspace methods. They appear to be more robust than classical iterative schemes and usually have better convergence properties.

3.2.1 Krylov subspace methods

If we replace $\boldsymbol{N}$ by $\boldsymbol{M} - \boldsymbol{A}$ in the iterative method (3.3), then it can be written as

$$x_{k+1} = x_k + \boldsymbol{M}^{-1} r_k, \tag{3.4}$$

where $r_k = b - \boldsymbol{A}x_k$ is the residual. If we start with x_0, the next steps can written as

$$x_1 = x_0 + \boldsymbol{M}^{-1} r_0,$$

$$x_2 = x_1 + \boldsymbol{M}^{-1} r_1,$$

substituting x_1 from the previous step and using $r_1 = b - Ax_1$, this leads to

$$x_2 = x_0 + 2\boldsymbol{M}^{-1} r_0 - \boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{M}^{-1} r_0$$

.
.
.
This implies that

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

The subspace $K_k(\boldsymbol{A}; r_0) := span\{r_0, \boldsymbol{A}r_0, ..., \boldsymbol{A}^{k-1} r_0\}$ is called the Krylov subspace of dimension k corresponding to matrix $\boldsymbol{A}$ 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 the system matrix.

The Krylov subspace is defined by its basis $v_1, v_2, ..., v_j$. This basis can be computed by the Arnoldi [3] algorithm. We start with $v_1 = r_0/\|r_0\|_2$, then compute $\boldsymbol{A}v_1$, make it orthogonal to v_1 and normalize it, to get 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 $K_j(\boldsymbol{A}; r_0)$. This basis is expanded by computing $w = \boldsymbol{A}v_j$ and orthonormalized with respect to the previous basis. The most commonly used algorithm is Arnoldi with the modified Gram-Schmidt procedure as shown in Algorithm 3.1, [40]. Let the matrix V_j be given as

$$V_j = [v_1 v_2, ..., v_j] \text{ where } span(v_1, v_2, ..., v_j) = K_j.$$

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

$$\boldsymbol{A}V_{m-1} = V_m H_{m,m-1}.$$

The $m \times (m-1)$ matrix $H_{m,m-1}$ is upper Hessenberg, and its elements $h_{i,j}$ are defined by Algorithm 3.1 known as the Arnoldi algorithm.

The Arnoldi algorithm is composed of matrix-vector products, inner products and vector updates. If $\boldsymbol{A}$ is symmetric, then $H_{m-1,m-1} = V_{m-1}^T \boldsymbol{A} V_{m-1}$ is also symmetric and tridiagonal. This leads to a three term recurrence in the Arnoldi process. Each new vector has only to be orthogonalized with respect to two previous vectors. The algorithm is known as the Lanczos algorithm. Krylov subspace methods are developed on the bases of these algorithms. For more details see [64], [83]. We will discuss some of the popular Krylov subspace methods that are used in numerical simulations.

Algorithm 3.1 Arnoldi algorithm with modified Gram-Schmidt procedure

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

Conjugate gradient method (CG)

For symmetric and positive definite systems, CG is the most effective Krylov method. CG finds a solution in a Krylov subspace such that

$$\|x - x_i\|_{\boldsymbol{A}} = \min_{y \in K^i(\boldsymbol{A}; r_0)} \|x - y\|_{\boldsymbol{A}} ,$$

where $(x, y)_{\boldsymbol{A}} = (x, \boldsymbol{A}y)$. The solution of this minimization problem leads to the conjugate gradient method [41].

Algorithm 3.2 Conjugate gradient method

$k = 0, x_0 = 0, r_0 = b$
While $r_k \neq 0$
    $k = k + 1$;
    If $k = 1$ do,
        $p_1 = r_0$;
    Else
        $\beta_k = \frac{r_{k-1}^T r_{k-1}}{r_{k-2}^T r_{k-2}}$
        $p_k = r_{k-1} + \beta_k p_{k-1}$
    End If
    $\alpha_k = \frac{r_{k-1}^T r_{k-1}}{p_k \boldsymbol{A} p_k}$
    $x_k = x_{k-1} + \alpha_k p_k$
    $r_k = r_{k-1} - \alpha_k \boldsymbol{A} p_k$
End While

From Algorithm 3.2, it is clear that the vectors from the previous iterations can be overwritten and only the vectors x_k, r_k, p_k and matrix $\boldsymbol{A}$ need to be stored. If $\boldsymbol{A}$ is dense, then matrix-vector multiplication costs N^2 operations, and the total cost is

$O(N^3)$, the same as for the direct methods. However, for sparse matrices the matrix-vector multiplication is much cheaper than $O(N^2)$, and in those cases CG may be more efficient. The convergence speed of CG depends on the condition number of the matrix.

Theorem 3.2.2. *The iterates obtained from the CG algorithm satisfy the following inequality [64]:*

$$\|x - x_k\|_{\boldsymbol{A}} \leq 2\left(\frac{\sqrt{K_2(\boldsymbol{A})} - 1}{\sqrt{K_2(\boldsymbol{A})} + 1}\right)^k \|x - x_0\|_{\boldsymbol{A}} \,. \tag{3.5}$$

This theorem suggests that CG is a linearly convergent process. However, it has been shown that if extremal eigenvalues are well-separated, superlinear convergence is observed [81]. It seems that after some iterations the condition number is replaced by a smaller effective condition number.

Bi-CGSTAB

Bi-CGSTAB [82] is a member of the family of Bi-conjugate gradient (Bi-CG) [36] algorithms. If the matrix $\boldsymbol{A}$ is symmetric and positive definite then the CG algorithm converges to the approximate solution. The CG method is based on the Lanczos algorithm. For nonsymmetric matrices the Bi-CG algorithm is based on Lanczos biorthogonalization. This algorithm not only solves the original system $\boldsymbol{A}x = b$ but also a linear system $\boldsymbol{A}^T x^* = b$. In the Bi-CG method, the residual vector can be written as $r_j = \phi_j(\boldsymbol{A})r_0$ and $\bar{r}_j = \phi_j(\boldsymbol{A}^T)\bar{r}_0$, where ϕ_j is a j^{th} order polynomial satisfying the constraint $\phi_j(0) = 1$. Sonneveld [74], observed that one can also construct the vectors $r_j = \phi_j^2(\boldsymbol{A})r_0$, using only the latter form of the innerproduct for recovering the bi-conjugate gradients parameters (which implicitly define the polynomials ϕ_j). This is the CGS method. In this method, the formation of vector $\bar{r}_j$ and multiplication by $\boldsymbol{A}^T$ can be avoided. However, CGS shows irregular convergence behavior in some cases. To remedy this difficulty Bi-CGSTAB (Bi-conjugate gradient stabilized) is developed. Bi-CGSTAB produces iterates with residual vectors of the form

$$r_j = \psi_j(\boldsymbol{A})\phi_j(\boldsymbol{A})r_0,$$

ψ_j is the new polynomial defined recursively at each step for stabilizing or smoothing the convergence.

The advantage of Algorithm 3.3 is that it is based on a short recurrence. It is always necessary to compare the norm of the updated residual to the exact residual as small changes in the algorithm can lead to instabilities.

GMRES

The generalized minimal residual algorithm is developed by Saad and Schultz [66]. This method is based on long recurrences and satisfies an optimality property. This means that it computes an approximation of the minimal of the residual. This method is used for nonsymmetric (non)singular matrices. GMRES is based on a modified

Algorithm 3.3 Bi-CGSTAB algorithm

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

Gram-Schmidt orthonormalization procedure and can optionally use a restart to control storage requirements. From Algorithm 3.4, it is clear that the Arnoldi algorithm is followed by a minimum least squares problem:

$$J(y) = \|b - \boldsymbol{A}x\|_2 = \|b - \boldsymbol{A}(x_0 + V_m y)\|_2$$

by using $r_0 = b - \boldsymbol{A}x$, $\boldsymbol{A}V_m = V_{m+1}\bar{H}_m$, $e_1 = [1, 0, ..., 0]^T$ 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 $x_m = x$ and one has reached the solution. It can be seen that the work per iteration and memory requirements increase for an increasing number of iterations. In order to avoid the problem of excessive storage requirements and computational costs for the orthogonalization, GMRES is usually restarted after m iterations which uses the last iteration as starting vector for the next restart. The restarted GMRES is denoted as GMRES(m). Unfortunately it is not clear what a suitable choice of m is. 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 [85].

Algorithm 3.4 GMRES algorithm

1. Compute $r_0 = b - \boldsymbol{A}x_0, \beta := \|r_0\|_2$, and $v_1 = r_0/\beta$
2. For $j = 1$ to m
3. Compute $w = \boldsymbol{A}v_j$;
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 loop
9. $v_{j+1} = w_j/h_{j+1,j}$
10. End
11. Define the $(m + 1)$xm 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 $x_m = x_0 + V_m y_m$

Theorem 3.2.3. *Suppose that $\boldsymbol{A}$ is diagonalizable so that $\boldsymbol{A} = X\Lambda X^{-1}$ 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 $C_c \in \mathbb{R}$ with $C_c > 0$ and having radius R_c with $C_c > R_c$, then $\epsilon^{(m)} \leq (\frac{R_c}{C_c})^m$ [66].

GMRESR

This method is a variant of GMRES developed by Vuik and van der Vorst [84]. The idea is that the GMRES method can be effectively combined with other iterative schemes. The outer iteration steps are performed by GCR [30], while the inner iteration steps can be performed by GMRES or with any other iterative method. In GMRESR, inner iterations are performed by GMRES. In Algorithm 3.5, if $m = 0$, 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 have been discussed in [84]. In some cases, when the iterative solution is close to the exact solution (i.e satisfies the stopping criterion), the m inner iterations of GMRES at that point will lead to a higher accuracy which is not required at that point. So it is never necessary to solve the inner iterations more accurately than the outer one [84].

In the next chapters, we will also use GCR to solve linear systems. The rate of convergence of GMRES and GCR are comparable. Like GMRES, GCR can also

Algorithm 3.5 GMRESR algorithm

1. x_0 is an initial guess and $r_o = b - \boldsymbol{A}x_0$
2. For $j = 1, 2, 3...$
3. $s_i = P_{m,i-1}(\boldsymbol{A})r_{i-1}$,
   s_i be the approximate solution of $\boldsymbol{A}s = r_{i-1}$
   obtained after m steps of an iterative method
4. $v_i = \boldsymbol{A}s_i$
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

be restarted if the required memory is not available. Another strategy known as *the truncation method* has a better convergence than the restart strategy, so if restarting or truncation is necessary truncated GCR is in general better than restarted GMRES. For properties and convergence results we refer to [30].

IDR(s)

IDR(s) (Induced dimension reduction) is a Krylov subspace method developed recently by Van Gijzen and Sonneveld [75] and is based on the principles of the IDR method which was proposed by Sonneveld in 1980. IDR(s) is a finite termination (Krylov) method for solving nonsymmetric linear systems. IDR(s) generates residuals $\boldsymbol{r}_n = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_n$ that are in subspaces $\mathcal{G}_j$ of decreasing dimension.
These nested subspaces are related by

$$\mathcal{G}_j = (\boldsymbol{I} - \omega_j \boldsymbol{A})(\mathcal{G}_{j-1} \cap \mathcal{S})$$

where

- $\mathcal{S}$ is a fixed proper subspace of $\mathbb{C}^N$. $\mathcal{S}$ can be taken to be the orthogonal complement of s randomly chosen vectors $\boldsymbol{p}_i, i = 1 \cdots s$.
- The parameters $\omega_j \in \mathbb{C}$ are nonzero scalars.

The parameter s defines the size of a subspace of search vectors. The larger s, the more memory is required. IDR(s) requires $N + N/s$ matrix-vector multiplications to get the exact solution. Theoretically Bi-CGSTAB gives the exact solution in $2N$

Algorithm 3.6 IDR(s) algorithm

1. While $\|\boldsymbol{r}_n\| > TOL$ or $n < MAXIT$
2. For $k = 0$ to s
3. Solve $\boldsymbol{c}$ from $\boldsymbol{P}^H d\boldsymbol{R}_n \boldsymbol{c} = \boldsymbol{P}^H \boldsymbol{r}_n$
4. $\boldsymbol{v} = \boldsymbol{r}_n - d\boldsymbol{R}_n \boldsymbol{c}$; $\boldsymbol{t} = \boldsymbol{A}\boldsymbol{v}$;
5. If $k = 0$
6. $\omega = (\boldsymbol{t}^H \boldsymbol{v})/(\boldsymbol{t}^H \boldsymbol{t})$;
7. End If
8. $d\boldsymbol{r}_n = -d\boldsymbol{R}_n \boldsymbol{c} - \omega \boldsymbol{t}$; $d\boldsymbol{x}_n = -d\boldsymbol{X}_n \boldsymbol{c} + \omega \boldsymbol{v}$;
9. $\boldsymbol{r}_{n+1} = \boldsymbol{r}_n + d\boldsymbol{r}_n$; $\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + d\boldsymbol{x}_n$;
10. $n = n + 1$;
11. $d\boldsymbol{R}_n = (d\boldsymbol{r}_{n-1} \cdots d\boldsymbol{r}_{n-s})$; $d\boldsymbol{X}_n = (d\boldsymbol{x}_{n-1} \cdots d\boldsymbol{x}_{n-s})$;
12. End For
13. End While

matrix-vector multiplications, provided exact arithmetic is used. IDR(1) has the same properties as Bi-CGSTAB. A disadvantage of Bi-CGSTAB is its erratic convergence behavior. For $s > 1$ the IDR(s) becomes more stable. The number of matrix-vector multiplications per iteration is equal to s, the number of iterations usually decreases for increasing s. The reduction of the number of iterations for increasing s is not monotonic. Large values of s sometimes even do not improve performance of IDR(s) [75]. Usually s is taken in the order of 4.

Multigrid

Multigrid methods are the most effective methods for solving large linear systems associated with elliptic PDEs. The idea of multigrid is based on a combination of two principles. First, the high frequency components of the error are reduced by applying a classical iterative method like a Jacobi or a Gauss Seidel scheme. These schemes are called smoothers. Next, low frequency error components are reduced by a coarse grid correction procedure. The smooth error components are represented as a solution of an appropriate coarser system. After solving the coarser problem, the solution is interpolated back to the fine grid to correct the fine grid approximation for its low frequency errors. The way multigrid components, i.e., smoothing, restriction, prolongation, and solution of the error equation on the coarse grid are linked to each other are shown in Algorithm 3.7.

Algorithm 3.7 is also known as the 2-grid algorithm; Step 4 can be optimized in various ways. For example, the error equation on the coarse grid is seldom solved exactly in practice. The customary method of solving it employs recursive calls to the 2-grid algorithm. If the recursion is carried out in a loop, thereby allowing different numbers of iterative sweeps on different coarse grids, we obtain the different V, W,

Algorithm 3.7 Solve $A_h u_h = b_h$

where subscript h is used for the fine grid and H for the coarse grid.

1. Perform smoothing by using k iterations of an iterative method (Jacobi, Gauss Seidel, etc) on the problem $\boldsymbol{A}_h u_h = b_h$
2. Compute the residual $r_h = b_h - \boldsymbol{A}_h u_h$
3. Restrict the residual $r_H = R r_h$
4. Solve for the coarse grid correction, $A_H e_H = r_H$
5. Prolongate and update $u_h = u_h + P e_H$
6. Perform smoothing by using l iterations of an iterative method (Jacobi, Gauss Seidel, etc) on the problem $\boldsymbol{A}_h u_h = b_h$

and F multigrid cycles. The multigrid components (as well as the cycle type) play an important role in achieving optimal convergence. It is widely accepted that the most efficient multigrid algorithm can be rendered by the *Full MultiGrid (FMG)* strategy. This involves predicting a solution of the fine grid equation by an interpolated version of the original equation (nor the error equation) on the coarse grid. The computational complexity of the FMG method is $O(N)$ and gives h-independent convergence. In geometric multigrid, restriction, prolongation, and coarse grids are chosen based on the geometric information. For more details see [78]. In case of absence of geometric data, an alternative known as algebraic multigrid (AMG) [16], [62], [63] can be employed.

AMG also uses these components; however, the information that travels from finer grid levels to coarse levels is not based on the geometric location of the grid points. To start the coarsening process, certain entries from matrix $\boldsymbol{A}_h$ are selected as influential in determining the solution. For example, if $a_{ij} \neq 0$ in $\boldsymbol{A}_h$, we say that point i in the grid is connected to point j and vice versa. The i^{th} row of the matrix then consists of only those entries that influence the unknown u_i. The influence of unknown u_j to u_i is said to be large if a small change in u_j gives a large change in u_i [18]. The influence of one unknown on another is decided by the corresponding coefficient. A coupling between two grid points i and j is strong if

$$|a_{ij}| > \theta \sqrt{a_{ii} a_{jj}},$$

where θ is a predefined coupling parameter [86]. A set of coarse variables is then defined by aggregating the nodes in the graph of strong connections using a greedy algorithm [86]. Once the coarse grid has been chosen, all operators in the coarse grid correction process, including the restriction and interpolation operators, are constructed based on information obtained from the coefficient matrix. Unlike multigrid, convergence of AMG does not require a robust smoothing strategy because the coarse grid correction process is designed to complement simple smoothers. A piecewise

constant interpolation operator $\hat{I}_H^h$ is defined that has positive nonzero entries of unity in positions determined so that its columns form a partition of unity over the aggregates. This tentative interpolation operator is then smoothed using Jacobi relaxation, defined by

$$I_H^h = (I - \omega D_h^{-1} \boldsymbol{A}_h^F) \hat{I}_H^h,$$

where ω is the relaxation parameter, $D_h = diag(\boldsymbol{A}_h^F)$ and $\boldsymbol{A}_h^F$ is the filtered matrix derived from $\boldsymbol{A}_h$ by adding all weak connections to the diagonal. The remainder of the multigrid components are formed based on the Galerkin condition [18], with restriction defined as $I_h^H = \left(I_H^h\right)^T$ and $A_H = I_h^H A m_h I_H^h$. This process is known as smoothed aggregation. AMG based on this interpolation technique shows nice convergence for problems with discontinuous coefficients and anisotropies [86], [76].

3.3 Preconditioning

Convergence of Krylov subspace method depends strongly on the spectrum of the coefficient matrix. Krylov methods show the best convergence if all eigenvalues are clustered around 1 or away from zero. Unfortunately, not all PDE's give rise to the desired eigenvalue distribution. Therefore, some techniques are required that change the eigenvalue spectrum of the matrix. This is known as preconditioning. With preconditioning, instead of solving a system $\boldsymbol{A}x = b$, one solves a system

$$P^{-1}\boldsymbol{A}x = P^{-1}b,$$

where P is the preconditioner. A good preconditioner must have a variety of properties. First, the method applied to the preconditioned system should converge quickly. This generally means that $P^{-1}\boldsymbol{A}$ has a small condition number (close to 1). Secondly, it should be easy to solve systems of the form $Pz = r$. The construction of the preconditioner should be efficient in both time and space. A system can also be preconditioned by a right preconditioner (post conditioner) or a split preconditioner. These are defined by:

$$\boldsymbol{A}P^{-1}y = b,\ x = P^{-1}y \text{ (right preconditioner)}$$

and

$$P_1^{-1}\boldsymbol{A}P_2^{-1}y = P_1^{-1}b,\ x = P_2^{-1}y \text{ (split preconditioner).}$$

Preconditioners are not restricted to Krylov subspace methods only, but for those methods the use of a preconditioner is the most natural. Since the scope of this thesis is restricted to the incompressible Navier-Stokes problem and we aim to solve the problem with Krylov subspace methods, we discuss preconditioners for the incompressible Navier-Stokes problem in detail in the next chapters.

3.4 Summary

In this chapter, various solution techniques have been discussed that can be used to solve a linear system. Direct solution methods give the exact solution at the cost of memory and CPU time. An alternative is to use iterative methods, which solve linear systems with low cost and memory up to a desired accuracy. Classical iterative methods based on the splitting of the coefficient matrix are easy to implement. They converge for certain classes of matrices. Krylov subspace methods based on matrix-vector multiplications give convergence for a wide range of problems. In absence of round off errors, Krylov methods give convergence in at most N iterations. However, if N is large these methods become expensive. Since convergence of Krylov subspace method depends on the eigenvalue spectrum, convergence is enhanced with some preconditioning technique. We discussed multigrid techniques that scale linearly with the number of unknowns if suitable components are used. Since classical iterative methods reduce high frequency errors efficiently, they are used as smoother in multigrid techniques. Multigrid can also be used for approximate preconditioner solves during a Krylov subspace iteration. In this usage, a one or a few cycles of MG usually suffice.

In the next chapter, we give an overview of some preconditioners that are used to solve the incompressible Navier-Stokes problem.

Chapter 4

Overview of Preconditioners

In this chapter preconditioners for the incompressible Navier-Stokes problem that accelerate Krylov subspace methods are discussed. In general, preconditioning techniques based on algebraic and physics-based approaches are widely used for the Navier-Stokes equations. Algebraic-type preconditioners are based on an ILU factorization or an approximate inverse of the coefficient matrix. These preconditioners are built on information available in the coefficient matrix. For the class of M-matrices, the importance of ILU as preconditioner was first highlighted by Meijerink and van der Vorst [52]. Later on, preconditioners were used in solving systems that arise from discretization of various PDE's [54, 49, 29, 28, 51, 12, 7], and references therein. The basic idea behind the ILU preconditioners is the same, but a variety of schemes have been developed to make ILU factors accurate and stable, especially for the indefinite systems. This is done by using various dropping strategies, scaling, reordering and pivoting techniques [14, 53, 1, 65].

Although, most of the ILU preconditioners present in the literature can be used to solve the incompressible Navier-Stokes problem, problem may arise due to the presence of zeros on the main diagonal. This may result in zero pivots during the ILU construction. Therefore, a dedicated efficient ILU preconditioner is required that can handle the zero block properly to avoid breakdown of ILU. In [25, 93, 94] reordering techniques are used to avoid zero pivots during elimination.

A special class of preconditioners is of the block structured type. The preconditioners are based on a (block) splitting of the matrix in a velocity and a pressure part. During each iteration each of the subblocks is solved separately. An important aspect of this approach is a good approximation of the Schur complement. The final goal is to get convergence independent of mesh size and Reynolds number. Various cheap approximations of the Schur complement matrix have been published. For an overview of this type of preconditioners, we refer to [33, 10, 31, 38, 43, 26, 58, 89, 47, 48, 7, 9, 35]. Some of those, which are used in combination with a block triangular preconditioner are discussed in this chapter.

4.1 ILU-type preconditioners

The idea of an incomplete LU (ILU) preconditioner stems from the use of a direct solver in which the system matrix is factorized into LU matrices. Since our system matrix is sparse in structure, LU factorization gives rise to dense factors which is not practical to solve by a direct solver due to memory requirement and work. However, an approximation of LU factors, known as incomplete LU factorization, can be an option in iterative methods. The idea of ILU is that some entries from LU are discarded at the time of formation of incomplete LU factors. The result is

$$\boldsymbol{A} = \hat{L}\hat{U} - R, \tag{4.1}$$

where $\hat{L}$ is incomplete lower triangular matrix, $\hat{U}$ is incomplete upper triangular matrix and R consists of dropped entries.

Definition 4.1.1. **(LU factors)** *In LU decomposition, a lower triangular matrix is denoted by $L = (l_{ij})$, such that $l_{ij} = 0$ if $i < j$, and $U = (u_{ij})$ as an upper triangular matrix, such that $u_{ij} = 0$ for $i > j$. $\mathbf{S}_n$ consists of all the pairs of indices of off-diagonal matrix entries, where $\mathbf{S}_n \equiv \{(i, j) | \ i \neq j, \ 1 \leq i \leq n, \ 1 \leq j \leq n\}$.*

In ILU decomposition, the pairs of indices $\mathbf{S} \subset \mathbf{S}_n$, consists of indices based on the dropping scheme. In ILU(0), $\mathbf{S}$ consists of only those indices where $a_{ij} \neq 0$.
The idea of ILU factorization was first developed for M-matrices.

Definition 4.1.2. **(M-matrix)** *The matrix $\boldsymbol{A} = (a_{ij})$ is an M-matrix if $a_{ij} \leq 0$ for $i \neq j$, the inverse $\boldsymbol{A}^{-1}$ exists and has positive elements $(\boldsymbol{A}^{-1})_{ij} \geq 0$.*

For this class of matrices, the incomplete LU decomposition is a regular splitting. By regular splitting we mean that if the decomposition given in (4.1) is applied in a classical iterative method settings,

$$\hat{L}\hat{U}x_{i+1} = Rx_i + b \text{ for every choice } x_0, \tag{4.2}$$

it will converge.

Theorem 4.1.1. *If $\boldsymbol{A} = (a_{ij})$ is an M-matrix of order $n \times n$, then there exists for every $\mathbf{S} \in \mathbf{S}_n$ a lower triangular matrix $\hat{L} = (l_{ij})$, with unit diagonal ($l_{ij} = 1$), an upper triangular matrix $\hat{U} = (u_{ij})$ and a matrix $R = (r_{ij})$ with*

$$\begin{aligned} l_{ij} &= 0 \ \ \textit{if } (i, j) \notin \mathbf{S}, \\ u_{ij} &= 0 \ \ \textit{if } (i, j) \notin \mathbf{S}, \\ r_{ij} &= 0 \ \ \textit{if } (i, j) \in \mathbf{S}, \end{aligned} \tag{4.3}$$

such that the splitting $\boldsymbol{A} = \hat{L}\hat{U} - R$ is regular. The factors $\hat{L}$ and $\hat{U}$ are unique.

Algorithm 4.1 ILU(0) factorization of matrix $\boldsymbol{A}$

```
1  For k = 1, ...., n − 1
2    For i = k + 1, n and if (i, k) ∈ S
3      a_ik = a_ik/a_kk
4       For j = k + 1, ...., n and for (i, j) ∈ S
5         a_ij = a_ij − a_ik ∗ a_kj
6       End For
7    End For
8  End For
```

For proof, see [52].

The creation of ILU(0) is defined in Algorithm 4.1. The upper part of $\boldsymbol{A}$ in Algorithm 4.1 now contains $\hat{U}$ (including diagonal) and lower part $\hat{L}$ has unit main diagonal. This is known as dropping by position strategy and S consists of only those indices which are a priori selected to place nonzeros entries in incomplete LU factors. Choice of indices can be based on matrix structure or connectivity of the grid points. The memory and work required is known beforehand. ILU preconditioners are very simple to implement and are quite effective for PDEs leading to M-matrices and diagonally dominant matrices.

If a matrix is symmetric and positive definite (SPD) then the Cholesky factorization exists. In incomplete Cholesky (IC) factorization $\boldsymbol{A}$ is decomposed in incomplete LL^T factors. Compared to M-matrices, IC factorization may breakdown due to zero or negative pivots in the SPD matrices. IC factorization for SPD matrices can be improved by increasing the matrix diagonal dominance, or by computing the preconditioner by using an approximation $\hat{\boldsymbol{A}}$ of $\boldsymbol{A}$ which is changed into an M-matrix [1], [49]. Of course, there are many schemes in the literature to improve performance of the IC factorization [7].

Example 4.1.1. (A diffusion problem) *We solve a diffusion problem on a square mesh using linear FEM elements (*64×64*) with suitable boundary conditions. This gives rise to an M-matrix. We employ the classical iterative method (4.2) and IC preconditioned CG. Figure 4.1 refers to the convergence plot that shows that IC decomposition is helpful in both classes of iterative methods. The classical iterative method converges and the convergence of CG improves if we use the IC factors as preconditioner.*

4.1.1 ILU for a general matrix

The ILU (IC) factorization is developed for M-matrices, however, it appears that it also works well for many applications where the coefficient matrix is not an M-matrix. The choice of fill-in is not restricted only to ILU(0). For more difficult problems, ILU(0) may give an approximation of $\boldsymbol{A}$ that is inaccurate and unable to produce a

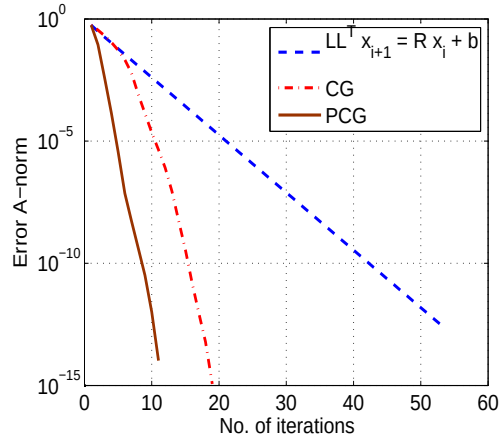


Figure 4.1: Convergence plot for diffusion problem solved with two different class of solvers (64×64 Q1 grid).

fast converging scheme [21]. In such a case, ILU(k) with extra fill-in, and ILU(t) with dropping based on size can do a better job than ILU(0).

Just like SPD matrices, indefinite problems also suffer from small or zero pivots. Besides that, unstable LU factors can arise in nonsymmetric problems. In case of an M-matrix, the properties of the matrix itself guarantee proper pivots and stable factors. In case of indefinite problems, use of ILU as blackbox preconditioner is not a good choice. The reason is that for indefinite problems, due to ill-conditioning, extremely small diagonal elements can seriously hamper the factorization.

Zero or small pivots can be avoided by various techniques ([7] and references therein). Even the formation of ILU factors without any breakdown does not guarantee a good quality of the preconditioner. The accuracy and stability are the proper indicators to judge the quality of a preconditioner. In case of an M-matrix, the ILU(0) preconditioner shows nice convergence. Introducing extra fill-in reduces $\|R\|_F$ (F stands for Frobenius norm) usually, which improves the convergence of the ILU preconditioned Krylov method. However, extra fill-in makes one iteration of a Krylov method more expensive. In case of indefinite problems, increasing extra fill-in does not improve convergence monotonically. Also the dropping by position strategy does not give convergence all the time. Therefore, other schemes based on threshold dropping are employed in which, instead of position, elements are dropped during the Gaussian elimination based on their magnitude. An example of such method is ILUT (T in this case is for threshold) [64]. With this approach, relatively accurate incomplete LU factors can be computed. However, the nonzero positions in this case are computed dynamically and predefined memory can not be used. A variant of ILUT, known as ILUTP, uses a parameter P to limit the number of nonzero entries per row [14]. The choice of the threshold value and preserving symmetry are issues in such schemes.

In indefinite problems, R alone can not be used to judge the quality of the preconditioner. The quantity $\|(\hat{L}\hat{U})^{-1}R\|$ should be small [51]. Multiplying both sides of (4.1) by $(\hat{L}\hat{U})^{-1}$ gives

$$(\hat{L}\hat{U})^{-1}A = I - (\hat{L}\hat{U})^{-1}R. \tag{4.4}$$

This means that $(\hat{L}\hat{U})^{-1}A$ is close to the identity if $(\hat{L}\hat{U})^{-1}R$ is small in some norm. In diagonally dominant matrices, $(\hat{L}\hat{U})^{-1}$ is well-conditioned, however, $(\hat{L}\hat{U})^{-1}$ may have a very large norm if the original matrix is not diagonally dominant, causing an increase in the overall norm of $(\hat{L}\hat{U})^{-1}R$. This will give rise to large perturbations in the identity matrix in (4.4). It is well-known that the performance of the ILU preconditioner can be improved with proper use of pivoting, reordering, scaling, and the use of a shifted diagonal [28, 64, 53].

Pivoting strategies can be used apriori or partially at the time of formation of ILU factors. In sparse matrices the profile and bandwidth of the matrices are important. They govern the efficiency of using the ILU factors. If the pivoting strategy does not increase the profile or bandwidth of the system, then pivoting can be effective, otherwise the search for a suitable pivot can make the preconditioner less efficient due to an increase in the profile of the matrix. Moreover, a large data structure that keeps information of permutation matrices at each step of ILU factorization makes the scheme more costly. The alternative to pivoting is apriori numbering. A suitable apriori renumbering improves the profile and bandwidth of the matrix and produces ILU factors that accelerate the Krylov method.

In [44, 71] an example of random matrices is given in which bandwidth reduction does not influence the convergence of ILU preconditioned Krylov methods. However, in [93, 25, 52, 28, 29, 51, 11, 14] it is shown that the combination of suitable node-renumbering techniques with ILU preconditioners improves convergence considerably for many PDE's resulting from engineering problems.

4.2 Application of ILU to Navier-Stokes

As discussed earlier, straightforward application of an ILU preconditioner is not effective or even not possible in some cases. This is true, for example, in the case of Navier-Stokes problem which has a large number of zeros on the main diagonal. The zero pivot problem may occur in the course of ILU operation of a saddle point type matrix unless proper care is taken. Moreover, in the Navier-Stokes problem the performance of a preconditioned Krylov iterative solver strongly depends on how variables are ordered in the resultant global matrix. Therefore, reordering is important for the effective ILU preconditioning of an assembled matrix. In the example below, we produce a globally reordered matrix by a reordering scheme, which minimizes bandwidth of the global matrix and avoids zero pivots.

Test Case 1. ***Driven cavity problem:*** *A 2D driven cavity problem is simulated in a square cavity* $(-1,\ 1) \times (-1,\ 1)$ *with enclosed boundary conditions with a lid moving*

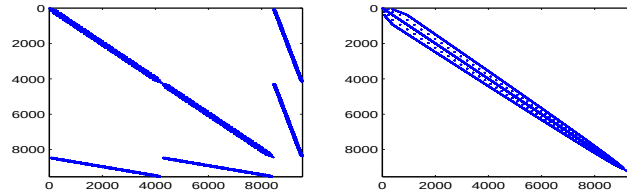


Figure 4.2: Test Case 1 discretized on 32×32 Q2-Q1 grid: Navier-Stokes matrices before (p-last) and after reordering.

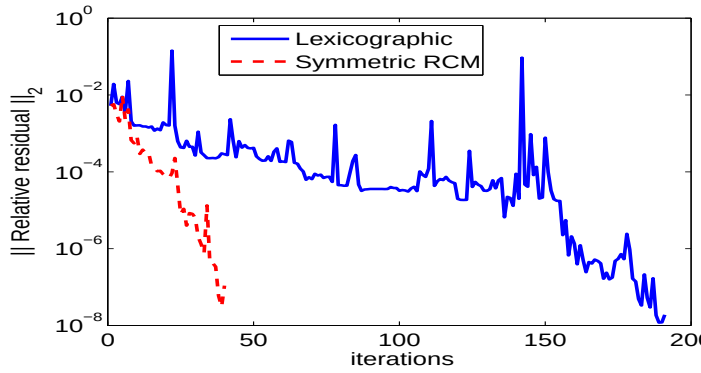


Figure 4.3: Test Case 1 discretized on 32×32 Q2-Q1 grid: Convergence curve of ILU preconditioned Bi-CGSTAB with $Re = 200$.

from left to right with speed:

$$u_x = 1 - x^4 \text{ at } y = 1;\ -1 \leq x \leq 1,$$

known as regularized cavity problem. The velocity and the pressure solution for the driven cavity problem is shown in Figure 4.4.

Example 4.2.1. *Test Case 1 is solved with ILU preconditioned Bi-CGSTAB with tolerance* 10^{-6}. *In Figure 4.2, it is visible that application of a reordering scheme (symmetric reverse Cuthill-McKee (RCM) from Matlab) reduces profile and bandwidth of the matrix. This reduces the number of nonzeros in* $\hat{L}\hat{U}$ *from 450k to 390k and the norm of R reduces from 0.16 to 0.13. Therefore, reduction in the number of iterations can be seen in Figure 4.3 (using LU1NC routine of Matlab).*

Blackbox ILU preconditioners like ILUPACK [14] enhance the convergence of the iterative method if used for the Navier-Stokes equations. However, convergence with ILU can be increased if the preconditioner is built on a matrix structure information. In the literature [25, 93], dedicated apriori reordering schemes for ILU-based Navier-Stokes solvers can be found. The schemes are helpful in accelerating the ILU preconditioned Krylov method. Since fill-in is based on the finite element structure, fill-in in these schemes is allowed at the pressure part that corresponds to the zero block in the global Navier-Stokes matrix. According to [25], two nodes are said to be coupled if they belong to the same finite element. Moreover, some work is also related to apriori pivoting in direct solvers to solve the incompressible Navier-Stokes problem [94].

We leave further discussion to Chapter 5, in which we present a new ordering technique for direct methods and ILU preconditioners.

4.3 Block preconditioners

We know that certain classes of preconditioners like ILU, AINV (approximate inverse) [11] are algebraic and they can be applied to the complete system by solving linear system(s) equal to the size of the problem. Besides that, there is a class of preconditioners that is applied in the form of subblocks for the velocity and pressure subsystems, separately. These preconditioners are known as block preconditioners. The preconditioners are based on a block factorization of the Navier-Stokes matrix. A block $\mathcal{L}_b\mathcal{D}_b\mathcal{U}_b$ decomposition of the coefficient matrix gives:

$$\mathcal{A} = \mathcal{L}_b\mathcal{D}_b\mathcal{U}_b = \begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} = \begin{bmatrix} I & 0 \\ BF^{-1} & I \end{bmatrix} \begin{bmatrix} F & 0 \\ 0 & S \end{bmatrix} \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{4.5}$$

where $S = -BF^{-1}B^T$ is the Schur complement matrix. Most preconditioners are based on a combination of these blocks and a suitable approximation of the Schur complement matrix. In this chapter, we will discuss preconditioners based on the $\mathcal{D}_b\mathcal{U}_b$ factors known as block triangular preconditioners:

$$P_t = \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix}. \tag{4.6}$$

The eigenvalues of the preconditioned system can be obtained from the generalized eigenvalue problem:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & B^T \\ 0 & S \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}. \tag{4.7}$$

From the first row one obtains

$$(1 - \lambda)(Fu + B^T p) = 0. \tag{4.8}$$

There are 2 possibilities: $1 - \lambda = 0$ or $(Fu + B^T p) = 0$.
If $1 - \lambda = 0$, then $\lambda = 1$ and we have eigenvalues equal to 1 of multiplicity n_u, otherwise

$$(Fu + B^T p) = 0 \text{ or } u = -F^{-1}B^T p. \tag{4.9}$$

Substituting (4.9) in the second row in (4.7) ($Bu = \lambda S p$) gives:

$$-BF^{-1}B^T p = \lambda S p. \tag{4.10}$$

This shows that if $S = -BF^{-1}B^T$ then $\lambda = 1$ has multiplicity n_p. An eigenvalue analysis suggests that GMRES converges in two iterations [55] if exact arithmetic is used. But in general, the use of F^{-1} and S^{-1} is not practical because they are very expensive to compute and to store. Usually, F^{-1} is formally approximated by a matrix $\hat{F}^{-1}$. Actually, such an approximation consists of a small number of iterations with an iterative method. The preconditioning steps involve solving $P_t z = r$, where $z = \begin{pmatrix} z_u \\ z_p \end{pmatrix}$ and $r = \begin{pmatrix} r_u \\ r_p \end{pmatrix}$ implies the steps given in Algorithm 4.2.

The preconditioner involves the solution of two subproblems associated with the

Algorithm 4.2 Perform $P_t z = r$

1 **Solve** $S z_p = r_p$
2 **update** $r_u = r_u - B^T z_p$
3 **Solve** $F z_u = r_u$

velocity and the pressure part. The approximation of the Schur complement is problem dependent. In general, the Schur complement matrix is not formed. Therefore, the approximate inverse $\hat{S}^{-1}$, is replaced by a simple spectral equivalent matrix such that the preconditioned matrix has a tightly clustered spectrum. Below we discuss a few preconditioners based on different approximations of the Schur complement matrix.

4.3.1 Approximate commutator based preconditioners

An effective approximation of the Schur complement matrix is developed by Kay, Login and Wathen ([43],[69]). This approximation utilizes the commutator action of two operators. The commutator of two operators x and y is defined as

$$[x,\ y] = xy - yx. \tag{4.11}$$

If x and y commute, then $[x,\ y] = 0$ i.e $xy = yx$.
The convection diffusion operator defined on the velocity space can be given as

$$\mathcal{L} = -\nu\nabla^2 + \mathbf{w_h}.\nabla, \tag{4.12}$$

where $\mathbf{w_h}$ is the approximation to the discrete velocity, computed in the most recent Picard iteration.

Pressure convection diffusion preconditioner

Suppose, that the commutator of the convection diffusion operator on the velocity space, multiplied by the gradient operator, on the velocity space, and the gradient operator, acting on the convection diffusion operator in the pressure space ($\mathcal{L}_p$), is small.

$$\varepsilon = \mathcal{L}\nabla - \nabla\mathcal{L}_p. \tag{4.13}$$

Then the discrete commutator in terms of finite element matrices given as

$$\varepsilon_h = (Q_v^{-1}F)(Q_v^{-1}B^T) - (Q_v^{-1}B^T)(Q_p^{-1}F_p), \tag{4.14}$$

might also be small. F_p is a discrete convection diffusion operator on pressure space. Q_v denotes the velocity mass matrix and Q_p the pressure mass matrix. The multiplication by Q_v^{-1} and Q_p^{-1}, transforms quantities from integrated values to nodal values. Pre-multiplication of (4.14) by $BF^{-1}Q_v$, post-multiplication by $F_p^{-1}Q_p$ and assuming that the commutator is small, leads to the Schur approximation

$$BF^{-1}B^T \approx BQ_v^{-1}B^TF_p^{-1}Q_p. \tag{4.15}$$

The expensive part $BQ_v^{-1}B^T$ in (4.15) is replaced by its spectral equivalent matrix A_p known as the pressure Laplacian matrix, so

$$S = -BF^{-1}B^T \approx -A_pF_p^{-1}Q_p. \tag{4.16}$$

The preconditioner (4.6), (4.16) is known as the Pressure Convection Diffusion (PCD) preconditioner. This preconditioner is also effective for stabilized finite elements without any adaptations of the Schur complement matrix approximation. The preconditioner has nice convergence properties especially for enclosed flows if the equations are linearized by a Picard method. The preconditioner gives rise to many iterations in inflow/outflow problems, the reason might be that an approximation of $BQ_v^{-1}B^T$ by A_p is well-defined only for enclosed flow problems [35]. As $\mathbf{w_h}$ in PCD is approximated by Picard linearization, so the linear system obtained from Picard requires fewer iterations than Newton. Boundary conditions are treated such that A_p and F_p are computed with Neumann boundary conditions for an enclosed flow problem. However, in outflow problems, rows and columns of A_p and F_p corresponding to the pressure nodes on an inflow boundary are treated as though they are associated with Dirichlet boundary conditions [35].

Test Case 1 is solved both with and without PCD preconditioner. Figure 4.5 shows that application of the preconditioner clusters the eigenvalues and thus enhances the convergence of the preconditioned iterative method (shown in Figure 4.6) . The preconditioner requires the action of a Poisson solve, a mass matrix solve and a matrix-vector product with F_p. From Table 4.1, it is clear the PCD gives mesh independent convergence. We use a direct solver to solve the velocity and the pressure subsystems or one V-cycle of MG is applied to F and A_p. The reason for the decrease in number of iterations with increase in the number of grid points is that the grid captures the features of the solution well as it become finer and finer. In Tables 4.1 and 4.2, we see

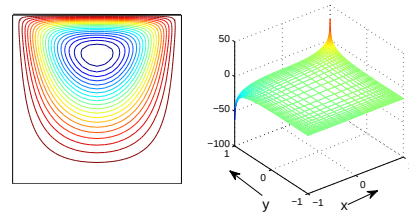


Figure 4.4: Equally spaced streamline plot (left) and presssure plot (right) of a Q2-Q1 approximation of 2D driven cavity flow problem with $Re = 200$.

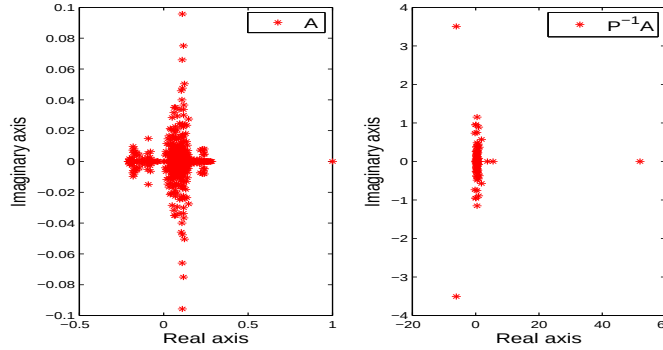


Figure 4.5: Eigenvalue of the original system and preconditioned with PCD.

a clear difference between iterations consumed by PCD using direct solver and MG. The small number of iterations with direct solver shows the dependence of PCD on subsystem accuracies. The number of iterations of PCD increases with the increase in Reynolds number as can be seen in Table 4.2.

Though PCD exhibits some nice convergence properties, the construction of two extra operators A_p and F_p makes PCD a difficult choice to use, specially in 3D due to extra memory requirements and extra boundary conditions. In [31], sparse approximate inverse technique is used to compute F_p algebraically. However, the scheme used in [31] reduces the effectiveness of the PCD preconditioners and iteration growth is observed compared to the previous approach, with increase in grid size and increase in Reynolds number.

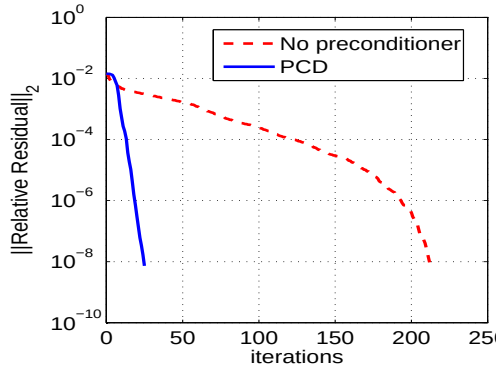


Figure 4.6: Convergence plot with PCD preconditioner.

Table 4.1: Number of PCD preconditioned Bi-CGSTAB iterations required to solve Test Case 1 with $Re = 100$ on different size grids.

Grid	Exact	MG
8×8	28	46
16×16	25	44
32×32	26	35
64×64	26	33

Table 4.2: PCD preconditioned Bi-CGSTAB iterations required to solve Test Case 1 on 64×64 grid.

Re	Exact	MG
100	20	29
200	26	35
300	28	38
400	34	45

Least squares commutator preconditioner

Another approach for the approximation of the Schur complement is described by Elman, Howle, Shadid, Shuttleworth and Tuminaro [31] and is known as a Least Squares Commutator (LSC) preconditioner. Instead of deriving the relation for the Schur complement, an approximation is made to the matrix operator, F_p, in (4.15), that makes the commutator small (4.14). This can be achieved by solving a least squares problem. For the *jth* column of matrix F_p, the least squares problem is of the form:

$$min\|[Q_v^{-1}FQ_v^{-1}B^T]_j - Q_v^{-1}B^TQ_p^{-1}[F_p]_j\|_{Q_v}, \tag{4.17}$$

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

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

which leads to the following definition of F_p:

$$F_p = Q_p(BQ_v^{-1}B^T)^{-1}(BQ_v^{-1}FQ_v^{-1}B^T).$$

Substituting this expression into (4.15) gives an approximation to the Schur complement matrix:

$$BF^{-1}B^T \approx (BQ_v^{-1}B^T)(BQ_v^{-1}FQ_v^{-1}B^T)^{-1}(BQ_v^{-1}B^T). \tag{4.18}$$

Usually, the inverse of the velocity mass matrix Q_v^{-1} is dense. In such a case the velocity mass matrix is replaced by the diagonal matrix $\hat{Q}_v$ (the main diagonal of Q_v). Without scaling, the Schur complement approximation in LSC is exactly the same as in the *BFBt* preconditioner [33]. However, it has been observed that the velocity mass matrix scaling increases the convergence rate of LSC preconditioner.

Algorithm 4.3 LSC preconditioner

1 **Solve** $S_f z_2 = r_2$ **Where** $S_f = B\hat{Q}_v{}^{-1}B^T$
2 **update** $r_2 = B\hat{Q}_v{}^{-1}F\hat{Q}_v{}^{-1}B^T z_2$
3 **Solve** $S_f z_2 = -r_2$
4 **update** $r_1 = r_1 - B^T z_2$
5 **Solve** $Fz_1 = r_1$

The Algorithm 4.3 involves two Poisson solves for the pressure subsystem and one velocity solve. The LSC preconditioner is build from readily available matrices and no extra boundary conditions are required.

Results in Table 4.3 show a mild increase in the number of iterations with the increase in number of grid points. According to [35] sometimes there is no h-dependency. The difference in the number of iterations obtained with the direct solver and MG is reduced with in increase in problem size. Table 4.4 shows a mild dependence of LSC on the Reynolds number. In [59] it is shown that for recirculating flows the convergence clearly depends on both issues. This is true because in construction of PCD and LSC, $\mathbf{w_h}$ is assumed to be constant for the commutator to be small. The preconditioner performs well for both enclosed and outflow problems.

Table 4.3: LSC preconditioned Bi-CGSTAB iterations required to solve Test Case 1 with $Re = 200$ on different size grids.

Grid	Exact	MG
8×8	14	25
16×16	15	22
32×32	17	21
64×64	25	26

Table 4.4: LSC preconditioned Bi-CGSTAB iterations required to solve Test Case 1 on 32×32 grid.

Re	Exact	MG
100	16	16
200	17	21
300	21	26
400	22	31

4.3.2 Augmented lagrangian approach (AL)

An effective preconditioner based on the augmented lagrangian approach is published by Benzi and Olshanskii [10]. This technique suggests a preconditioner of the form:

$$P_{AL} = \begin{bmatrix} F_\gamma & B \\ 0 & \hat{S} \end{bmatrix}, \tag{4.19}$$

where $F_\gamma = F + \gamma B^T W^{-1} B$ and the inverse of the Schur complement is approximated by

$$\hat{S}^{-1} = -(\nu \hat{Q}_p^{-1} + \gamma W^{-1}). \tag{4.20}$$

$\hat{Q}_p$ denotes the approximate pressure mass matrix, ν is the viscosity and $\gamma > 0$ is a parameter. Usually, W is also replaced by $\hat{Q}_p$. For constant pressure approximation, Q_p is a diagonal matrix. For a linear pressure approximation, Q_p is replaced by the spectrally equivalent diagonal matrix. For a diagonal matrix $\hat{Q}_p$, the computation of the inverse approximate Schur complement is very cheap. The preconditioner is known as AL preconditioner (P_{AL}). For this preconditioner, the original system given in (2.21) is replaced by

$$\begin{bmatrix} F_\gamma & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ 0 \end{bmatrix}. \tag{4.21}$$

Since $Bu = 0$, we can add the term $\gamma B^T W^{-1} Bu$ to the first row in (2.21) without any change in the right-hand side.

Theorem 4.3.1. *Assume that $W = Q_p$, then the preconditioned matrix has eigenvalue 1 of multiplicity n_u. The remaining n_p eigenvalues are given by*

$$\lambda_i = \frac{\gamma + \nu}{\gamma - \mu_i^{-1}}, \; 1 \leq i \leq n_p, \tag{4.22}$$

where μ_i satisfies the generalized eigenvalues problem $BF^{-1}B^T p = \mu Q_p p$ [10].

For proof see [10]. This means that for $\gamma \to \infty$, all eigenvalues go to 1. AL preconditioned Krylov subspace method will converge in 1 iteration, however, iterative solution of the velocity subsystem is no longer possible. This scheme is close to the penalty function formulation [24].

Results given in [10] reveal that the convergence of an iterative method with this preconditioner is independent of the mesh size and Reynolds number and seems robust. The method is efficient, if it is used as a preconditioner in solution of the Navier-Stokes problem, linearized by Picard's iteration for both constant and linear pressure approximations. But this preconditioner requires an efficient method to solve the velocity subsystem in (4.19) . The quality of the preconditioner is based on the following issues:

- The efficiency of the preconditioner depends on the additional term $\gamma B^T W^{-1} B$ in the velocity matrix. If the problem does not have cross derivatives, then F_γ introduces a coupling between components of the velocity vector, and thus has a greater number of nonzero entries than F irrespective of the discretization scheme used. Based on the discretization scheme used, the increase in the number of nonzeros is more for the elements that use continuous pressure approximations (Q2-Q1) than for elements that use discontinuous linear pressure approximations (Q2-P1).

- Usually γ is taken to be one. Moreover the convergence of the velocity subsystem solver depends on the value of γ. A large γ makes the system F_γ more ill conditioned and iterative methods fail to converge for such systems. Benzi [10] has used MG with a new smoothing technique to solve the velocity subsystem.

We perform numerical experiments with the AL preconditioner by solving Test Case 1. The pressure subsystem is just a diagonal scaling of the residual. The velocity subsystem is solved with a direct solver and ILU. If Q2-P1 discretization is employed an increase in the number of nonzeros from F to F_γ is almost two times whereas with Q2-Q1 discretization the increase is much higher. Increase in the number of nonzeros for the velocity subsystem is shown in Figure 4.7. With $\gamma = 1$ and greater, the solution of linear system $F_\gamma z_1 = r_1$ with ILU(0) preconditioned Krylov method is not possible. This is due to computation of unstable ILU factors. This arises if the matrix is far from diagonal dominance [21]. The stability of the ILU factors can be checked by $\|(LU)^{-1}e\|_\infty$ (refered to as condest in the Table 4.5) where e is a vector with all ones. It appears that this quantity is large e.g 10^8 for even small problems. Application of ILUT (in Matlab) makes the preconditioner very expensive due to a large increase in the number of nonzero entries and does not guarantee convergence. If we compute ILU of F_γ with the nonzero pattern of F the preconditioner becomes effective, at least for our problem. For example in Table 4.5, A1 represents the nonzero pattern with F and A2 with F_γ. It can been seen that skipping some entries gives stable ILU factors with small R and condest [79].

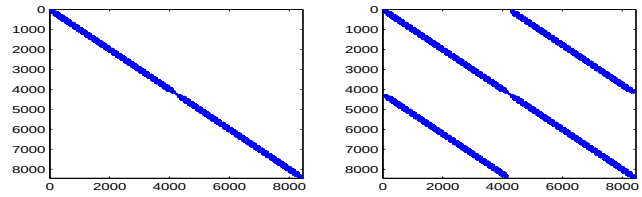


Figure 4.7: Nonzero pattern of the velocity matrix in 32×32 Q2-P1 driven cavity flow problem with $Re = 200$: F (left), F_γ (right).

Table 4.5: Analysis of ILU preconditioner of F_γ.

Grid(nonzero	$max(1/pivot)$	$max(L+U, F_\gamma)$	condest	$\|\|R\|\|_\infty$	Remarks
pattern)	Q2-P1				
8×24(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)
16×24(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)
32×24(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)

Table 4.6: AL preconditioned GCR iterations required to solve Test Case 1 with $Re = 200$ on different grids.

Grid	Exact	ILU
8×8	6	7
16×16	6	7
32×32	6	7
64×64	6	7

Table 4.7: AL preconditioned Bi-CGSTAB iterations required to solve Test Case 1 on 32×32 grid.

Re	Exact	ILU
100	6	7
200	6	7
300	6	7
400	6	7

The application of ILU(0) -with F_γ nonzero pattern- increases the norm of R and $(LU)^{-1}$, while ILUT gives rise to large number of nonzero entries.

Results in Table 4.6 indicate that the AL preconditioner gives convergence independent of grid size. Also, we do not see much difference in outer iterations whether the velocity subsystem is solved exactly or inexactly with an accuracy of 10^{-2}. Table 4.7 shows that AL convergence is independent of the Reynolds number.

Despite that the AL preconditioner has nice convergence, an effective solver is required to solve the linear system corresponding to F_γ, otherwise the gain in outer iterations is offset by the loss in solving the velocity subsystem. The choice of γ shifts the work between outer and inner iterations. If F is SPD, F_γ will also be SPD but it will lose diagonal dominance due to the addition of an ill conditioned matrix. A similar approach is also used in [26] known as AC (artificial incompressibility) and GD (Grad-Div) preconditioner. Compared to the AL preconditioner, AC and GD are more dependent on the choice of γ while with $\gamma = 1$, AL gives convergence in a fixed number of iterations for a wide range of problems and Reynolds numbers. For small values of γ, an inexact solver can be used to solve system $Fz_1 = r_1$. However, for γ of $O(1)$, an efficient solver is required.

4.3.3 Remarks on selection of preconditioner

From the discussion above it is hard to decide which method gives the best results for various flow problem domains, boundary conditions and discretization schemes. Even the developers of PCD and LSC are unable to conclude explicitly about the choice when to use these preconditioners. Moreover, we are more interested in comparing our preconditioners with one of the best preconditioners available in the literature. In this thesis, we choose LSC for our experiments because:

- LSC is built from readily available matrices. Unlike PCD, the Schur comple-

ment approximation does not require information of boundary conditions. An algebraic construction of F_p results in a decrease in effectiveness of the PCD preconditioner.

- PCD gives mesh independent convergence especially for enclosed flow problems, while LSC shows mild dependence on mesh size. However, LSC always converges in fewer iterations than PCD. In some cases, LSC shows convergence behaviour which is twice as good as compared to the convergence of the PCD preconditioner [31]. Moreover the dependence of LSC on mesh sizes reduces at high Reynolds numbers. It may happen that for a large problem, PCD performs better than LSC, however until now no problem is reported where PCD costs less iterations than LSC [35], [31].

- In terms of outer iterations, AL shows better convergence than PCD and LSC. However, the inner/outer iterations are dependent on parameter γ. An efficient and dedicated solver is required to solve the velocity subsystem in AL. Moreover, AL becomes expensive to use for Taylor-Hood elements.

4.4 Summary

In this chapter we discussed two classes of preconditioners that can be used to solve the incompressible Navier-Stokes problem. ILU-type preconditioners are based on an incomplete factorization of the coefficient matrix. These preconditioners are well-defined for PDEs the discretization of which gives rise to an M-matrix. For other type of PDEs it may be necessary to make ILU factors accurate and stable in order to get convergence. In Navier-Stokes direct application of ILU may breakdown due to zeros on the main diagonal. Therefore, some suitable renumbering scheme is required to make the factors effective. In the next chapter we will discuss such a scheme that can be used to solve the incompressible Navier-Stokes problem.

Besides, we discussed preconditioners based on block factors of the coefficient matrix. We discussed block triangular preconditioners that are based on some approximation of the Schur complement, the inexact solution of the velocity and the approximated Schur system.

We overviewed the most popular ones in combination with Krylov subspace methods, PCD, LSC and AL. PCD gives mesh independent -but Reynolds number dependent convergence. The construction of extra operators that are required for the Schur complement makes it expensive. LSC is built from available matrices. Though LSC shows mild dependence on mesh size and Reynolds number, its convergence is always better (two times in some cases) than PCD [31]. Therefore, in the remaining chapters we will compare our preconditioners with LSC.

In the AL preconditioner, an extra term is added both to the velocity matrix in the original problem as well as to the preconditioner. This enhances the convergence of the modified problem. The preconditioner gives mesh and Reynolds independent convergence for a long range of problems. However, convergence of AL depends on

the choice of parameter γ. To get convergence, an efficient MG solver for the (1,1) block in the preconditioner is required. In general, the demand of efficient MG solver and choice of γ restricts the choice to use AL.

Chapter 5

Saddle point ILU preconditioner

In this chapter, we propose a new, a priori, ordering of the unknowns, which in combination with a suitable renumbering of nodes, results in an optimal bandwidth or profile (envelope) of the coefficient matrix in the Navier-Stokes problem. The new ordering scheme avoids breakdown of *LU*/ILU factorization. In the direct method, it reduces profile and bandwidth of the matrix and thus reduces memory and CPU time. Moreover, if the scheme is applied prior to the construction of the ILU preconditioner, reduction in the number of iterations is observed with the reordered ILU preconditioner. Since the preconditioner is constructed for saddle point problems, we call it the *saddle point ILU (SILU)* preconditioner.

5.1 Ordering of the system

For an application of block preconditioners from standard finite element codes, adaptation of the matrix builder and solver is necessary, since splitting of velocity and pressure unknowns is required. From a practical point of view, it would be attractive, if standard classical iterative solution schemes, like preconditioned Krylov solvers, could be applied, without any changes. However, in the case of non-stabilized elements, the zero pressure block in the continuity equation, prevents straightforward application of *LU* and ILU factorization. If the common ordering of unknowns is used, i.e. placing first all unknowns of node 1, then those of node 2 and so on (Table 5.1), one might get a zero pivot, especially if velocities at some boundaries are prescribed and therefore both factorizations may fail. Pivoting, [8], on the other hand, will result in a large increase of memory usage and, as a consequence, computation time. Besides that, it is hard to predict, apriori, the amount of memory required, which from an implementation point of view is, not very practical. To avoid this problem, it is better to use a suitable apriori reordering of unknowns. As pointed out by Wille and others [25], [93],[94], pivoting is not necessary, when the unknowns are ordered sequentially, so that all velocity unknowns come first and then all the pressure unknowns; such as,

Table 5.1: Ordering of unknowns for 5 nodes grid.

p-last	node-wise
u_1	u_1
v_1	v_1
u_2	p_1
v_2	u_2
u_3	v_2
v_3	p_2
u_4	u_3
v_4	v_3
u_5	p_3
v_5	u_4
p_1	v_4
p_2	p_4
p_3	u_5
p_4	v_5
p_5	p_5

in the block preconditioners. The reason being that during (incomplete) factorization the zeros at the main diagonal will vanish, provided that fill-in is allowed, based on the connectivity of nodal points, rather than actual zeros in the matrix. Renumbering of nodal points as suggested by Wille, and also classical renumbering techniques as Cuthill-McKee (CMK) [23] and Sloan [73] may decrease the memory and computation time for direct solvers. An optimal numbering of unknowns, for a direct solver, usually improves the convergence of ILU preconditioned Krylov solvers, [29], [51], but exceptions to this rule exist [7].

5.1.1 Ordering used in direct method

If, for a direct solver, we use the same ordering as in case of the block preconditioners, i.e. placing first all velocity unknowns and then all pressure unknowns, we end up with a very large profile of the matrix. This is true even if we use an optimal node renumbering. The main advantage of this ordering is that no pivoting is necessary, since during factorization, the zeros on the main diagonal in the zero pressure block disappear, see for example [93]. In the remaining part of this chapter we shall refer to this reordering as *p-last*. Figure 5.1 shows an example of the nonzero structure of the matrix for the p-last ordering, applied to a 4×4 rectangular grid of Q2-Q1 elements, where a lexicographic numbering of the nodes is used.

A much smaller profile will be achieved, if we order the unknowns in the sequence of the nodal points. However, especially in the case of Dirichlet boundary conditions for the velocity, this may lead to zero pivots. So, if we try to avoid pivoting during elimination, this ordering can not be applied. In order to get a reordering that has almost the same favorable profile as the node-wise ordering, but does not lead to zero pivots, we need to define the concept of levels, which originates from the classical

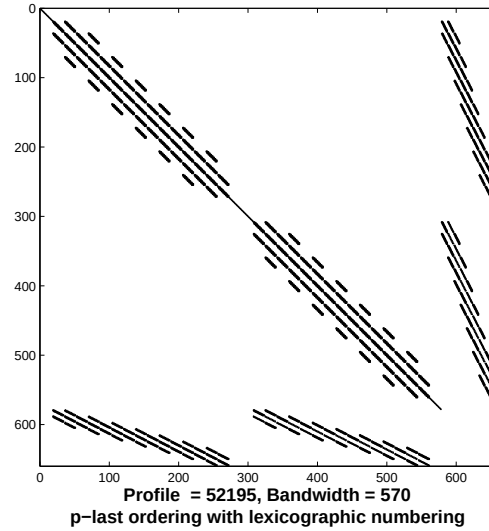


Figure 5.1: p-last ordering of unknowns of the Stokes matrix.

Cuthill-McKee renumbering scheme.

Let us first define the notion of levels for Cuthill-McKee. Suppose we have created levels 1 to i-1. Then level i is defined as the set of nodes that are connected directly to level i-1, and are not in one of the prior levels. Nodes are connected if they belong to the same element.

The first level may be defined as a point, or even a line in R^2 or a surface in R^3. This definition also applies in case of structured grids. For example in the 4×4 structured grid of Figure 5.2, with Q2-Q1 elements, the first level might consist of the nodes 1 to 9. Level 2 consists of the nodes 10 to 29 and so on. It is clear that nodes in level i are only connected to nodes in level i-1 and level i+1.

In case of a different renumbering scheme, like Sloan [73] or the one proposed by Wille et al [93], we define levels in the following way:

Suppose levels 1 to i-1 have been constructed. Let node, k, be the node with highest node number, that is directly connected to nodes in level i-1 Then level i, consists of node k, and all nodes with node numbers less than k but not belonging to one of the levels 1 to i-1. The first level is defined as node 1.

Once the levels have been defined, we reorder the unknowns in the following way. First we take all the velocities of level 1, then all pressures of level 1. Next we do the same for level 2, and repeat this process for all nodes. So instead of a global block ordering we apply a block ordering per level. Such an approach has two advantages.

First, the profile is hardly enlarged, compared to the optimal ordering, since the local band width is defined by the largest distance in node numbers.

Second, due to the local block reordering, zero pivots become nonzero, during factorization, and no a posteriori pivoting is required. In the remainder we shall refer

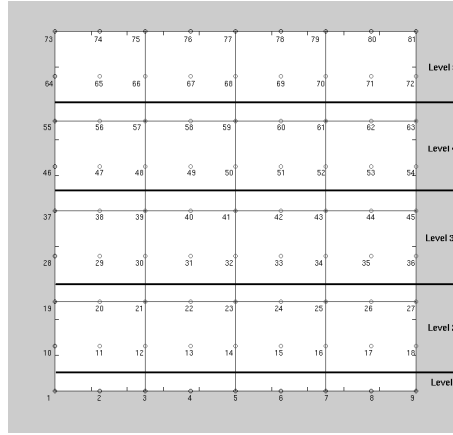


Figure 5.2: Levels defined for 4x4 Q2-Q1 grid.

to this ordering technique as *p-last per level*.

One has to be careful at the start of this process. If, for example, the velocities in node 1, are prescribed, we start with a pressure unknown that gives rise to a zero pivot. Therefore, we always combine levels 1 and 2, into a new level. If the number of free velocity unknowns in this new level is less than the number of pressure unknowns, we also add the next level to level 1, and if necessary this process is repeated. In practice combinations of 2 or 3 levels is sufficient. Note that the starting level always has a small contribution to the global profile. Figure 5.3 shows the effect of the p-last per level renumbering, combined with Sloan (A.1) and Cuthill-McKee (A.2) renumbering, respectively. The grid used, consists of 8×8 Q2-Q1 elements. The gain in memory is clear, even for this small example.

So our reordering technique p-last per level, in combination with a suitable node renumbering strategy, produces a nearly optimal profile and avoids the need for pivoting in case of direct solvers. It has been applied to many practical problems, without ever producing small pivots. Since optimal reordering of unknowns for direct methods, is usually also suitable for ILU preconditioners, we consider this method in the next subsection.

5.1.2 Application to ILU preconditioning

Since an optimal ordering of unknowns for a direct solver, usually improves the behavior of an ILU preconditioner, we investigate p-last per level ordering, as well as p-last ordering, in combination with ILU.

S in our case is, of fill-in positions as the set of unknowns, that are directly connected. This implies that, zeros in the pressure block, may also be part of the set **S**, provided that there is a connectivity with velocity unknowns. The ILU decomposition $\mathcal{A} \approx \hat{L}D^{-1}\hat{U}$ is defined as:

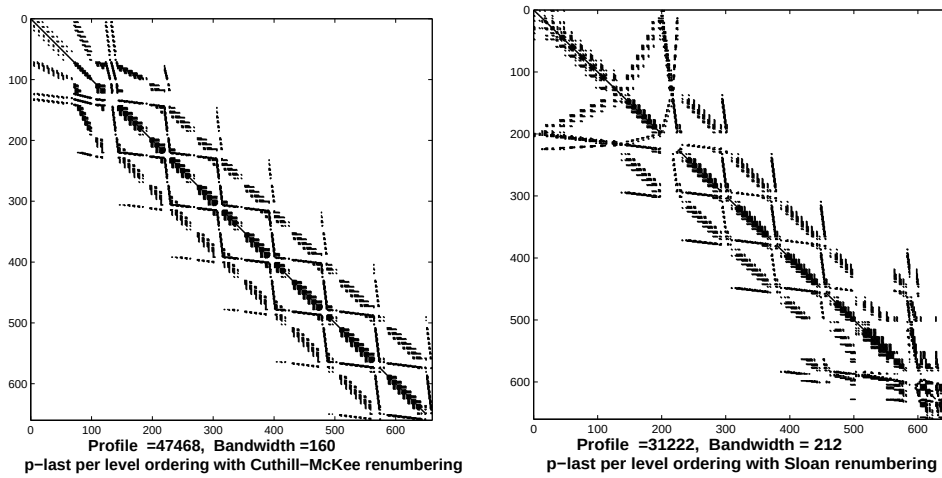


Figure 5.3: Effect of Sloan and Cuthill-McKee renumbering of grid points and p-last per level reordering of unknowns on the profile and bandwidth of the matrix.

Definition 5.1.1. *SILU*

1. $l_{i,j} = 0$ *for* $(i, j) \notin \mathbf{S}$,
2. $u_{i,j} = 0$ *for* $(i, j) \notin \mathbf{S}$,
3. $(\hat{L}D^{-1}\hat{U})_{i,j} = a_{i,j}$ *for* $(i, j) \in \mathbf{S}$.

The solution method consists of following steps:

1. After mesh generation the grid points are reordered by the Cuthill-McKee or Sloan renumbering method.
2. To prevent zero pivot during incomplete *LU* factorization, the unknowns per grid point are reordered by the p-last or p-last per level reordering methods.
3. An incomplete *LU* decomposition of the reordered matrix is constructed and used as a preconditioner.
4. A preconditioned Krylov subspace method (GMRES, Bi-CGSTAB, etc.) is used to approximate the solution.

Experiments in Section 5.2.2, show that in a large number of practical cases, this method performs very well. However, in some cases the Krylov method converges slowly, or even diverges, for example in case of stretched grids, using elements with a large aspect ratio. In that case, we apply *extra fill-in* referred to as ILUF. Extra fill-in is defined by adding all neighbor points of the standard ILU node structure to the connectivity set, provided these nodes do not affect the envelope of the original matrix. In many cases, extra fill-in solves the convergence problem, but at the cost of extra memory and computing time per iteration.

Lumping

If the off-diagonal components of the matrix have the same sign as the diagonal of matrix, then these components are added to the diagonal of the matrix and made zero themselves. Of course, this is only used to compute the preconditioner. Though this type of lumping is not defined for saddle point system, in our experience, the preconditioning matrix P_ℓ(lumped) improves convergence in some cases but it should only be used if one does not achieve convergence with extra fill-in.

Perturbation of the continuity equation

Convergence can be improved by perturbing the continuity equation with a factor ϵp. The discretized form of the perturbed system of equation is given as

$$\begin{bmatrix} F & B^T \\ B & \epsilon Q_p \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{5.1}$$

So the incompressibility condition is violated by a small amount, without influences the solution. Note that this is almost the same as applying the penalty function method [24]. The only difference is that p is not eliminated from the second row in (5.1). Since, in general, it is hard to find a suitable value of ϵ we have decided to use mostly $\epsilon = 0$. In some cases, stretched grids and Q2-P1 discretization, we have perturbed the incompressibility constraints and found good convergence.

5.1.3 Breakdown of *LU* or ILU factorization

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

To explain how the minimum size of the first level must be chosen we consider a 2×2 Q2-Q1, Taylor-Hood element subdivision of a square shown in Figure 5.4. If all the velocities at the boundary are prescribed, restricting the initial set to the (oblique) dashed region, i.e. nodes 1 to 7, implies that in set 1 we have only 2 unknown velocities and 4 unknown pressures. Even if we start with the velocities, Gaussian elimination in these rows will not remove all zeros on the diagonal. This is the same reason as we have to satisfy the LBB condition. Adding node 8 to the dashed region makes the number of velocity unknowns in the first level equal to the number of pressure unknowns and the problem no longer exists.

So on the first level we need at least the same number of unprescribed velocity degrees of freedom as there are pressure degree of freedom. Furthermore, the velocity unknowns should have a nonzero connection with the pressure unknowns. Experimentally, we have seen that this also holds for ILU preconditioner. Consider the

nonsymmetrical case where we multiply the discretized continuity equation by a minus sign, hence $-Bu = -g$. We will prove that the preconditioner exists theoretically for a ILUD (see Definition 5.1.2) decomposition of this matrix,

$$\mathcal{A}_{ns} = \begin{bmatrix} F & B^T \\ -B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ -g \end{bmatrix}, \tag{5.2}$$

in which the off-diagonal elements of $\hat{L}$ and $\hat{U}$ are taken equal to the corresponding elements in $\mathcal{A}_{ns}$. Only the matrix $\mathcal{D}$ has to be determined. In formulas:

Definition 5.1.2. ***ILUD:***

1. $diag(\hat{L}) = diag(\hat{U}) = \mathcal{D}$,
2. $l_{i,j} = a_{i,j}$ *for* $i > j$ *and* $u_{i,j} = a_{i,j}$ *for* $j > i$,
3. $(\hat{L}\mathcal{D}^{-1}\hat{U}L)_{i,j} = a_{i,j}$.

Proposition 5.1.1. *If we use the p-last ordering and assume that the ILUD decomposition of F exists with positive matrix D then the ILUD decomposition exists because every column of* B^T *contains a nonzero element . Note that if* B^T *has a zero column then* $\mathcal{A}_{ns}$ *is singular.*

Proof: Before the proof, for simplicity second row in (2.21) is multiplied with a minus one $(-B)$. We consider the computation of $\mathcal{D}$:

$$(\hat{L}\mathcal{D}^{-1}\hat{U})_{i,i} = d_i + \sum_{j=1}^{i-1} \frac{l_{i,j} \cdot u_{j,i}}{d_j} = a_{i,i}, \tag{5.3}$$

this will lead to

$$d_i = a_{i,i} - \sum_{j=1}^{i-1} \frac{a_{i,j} \cdot a_{j,i}}{d_j}. \tag{5.4}$$

From the assumption it follows that the ILUD decomposition of F exists and thus $d_j > 0$ for $j = 1, ..., n_u$. For $i \in (n_u + 1, N)$ we have $a_{i,j} = -a_{j,i}$ and $a_{i,i} = 0$. This together with (5.4) implies that $d_i = \sum_{j=1}^{i-1} \frac{a_{i,j}^2}{d_j}$. Since the norm of a column B^T is nonzero we have $\sum_{j=1}^{i-1} \frac{a_{i,j}^2}{d_j} > 0$. Combined with $d_k > 0$ for $k < i$ it follows that

$$d_i \geq (\min_{1 \leq k \leq i-1} \frac{1}{d_k}) \sum_{j=1}^{i-1} a_{i,j}^2 > 0. \tag{5.5}$$

Proposition 5.1.2. *For an arbitrary ordering we suppose that the ILUD decomposition exists for all* $j < i$*, where the* i^{th} *row is related to the continuity equation. If the* i^{th} *(pressure) unknown is preceded by at least one velocity unknown with a nonzero connection to this pressure unknown (so there is one* $k < i$ *such that* $a_{i,k} \neq 0$*), then the ILUD decomposition exists and* $d_i > 0$*.*

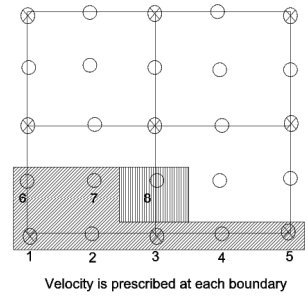


Figure 5.4: 2x2 Q2-Q1 grid.

Proof: It follows again from (5.4) that

$$d_i > \sum_{j=1}^{i-1} \frac{l_{i,j} \cdot u_{j,i}}{d_j}. \tag{5.6}$$

Since $d_j > 0$ for $j < i$ and $a_{i,k}^2 > 0$ for at least one $k < i$ we obtain $d_i > 0$.

From the above propositions it is clear that the ILUD decomposition gives positive diagonal elements. If we use the original matrix (symmetric case) the diagonal elements corresponding to the pressure part appear to be negative. This is also the case for ILU decomposition, but we have no theoratical proof.

Our reordering is applicable for all types of unstructured grids. The ordering introduced by Wille [93], seems to be very effective for Navier-Stokes problems on a square grid. It is not clear at this moment, how to generalize this to an unstructured grid. However, the main difference is that Wille renumbers the nodes and uses the p-last ordering, while we apply a node renumbering technique, which could be for example Wille's, along with reordering of unknowns by the introduction of levels.

5.2 Numerical experiments

In this section we present some numerical experiments. We start to use our renumbering scheme in a direct method. Later on, we investigate properties of the saddle point ILU preconditioner. SILU is tested for two benchmark problems, the lid driven cavity problem given in Test Case 1, and a backward facing step given as:

Test Case 2. ***Backward facing step problem:*** *The L shaped domain* $(-1, L) \times (-1, 1)$*, known as the backward facing step (see Figure 5.5). A Poiseuille flow profile is imposed on the inflow* $(x = -1;\ 0 \leq y \leq 1)$ *and no slip conditions are imposed on the side walls. Neumann conditions are applied at the outflow which automatically sets the mean outflow pressure to zero. The Navier-Stokes equations are also solved in a 3D backward facing step domain with parabolic inflow velocity profile, no slip condition on the side walls and Neumann conditions at the outflow.*

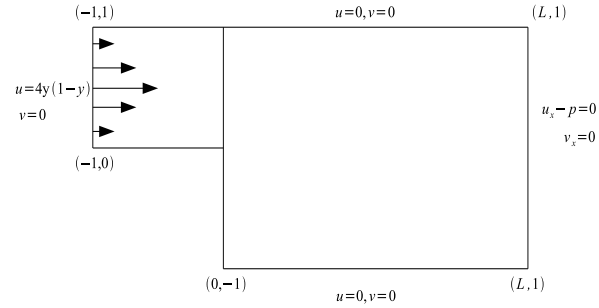


Figure 5.5: Backward facing step or L shaped domain.

5.2.1 Impact of reordering on the direct solver

In this section, we present some results to see how our reordering strategy effects the efficiency of the direct solver. We report our findings with our renumbering scheme. Our renumbering scheme effectively reduces the profile and bandwidth of the matrix. In Table 5.2, we see the reduction with the Sloan and Cuthill-McKee renumbering methods with p-last per level reordering of unknowns. Profile and bandwidth reduction are computed by dividing the p-last per level profile and bandwith by p-last. Profile reduction with the Sloan method is better than Cuthill-McKee, while in bandwidth reduction Cuthill-McKee performs better than Sloan. Thus, our reordering method reduces the memory and work and computation time if the system is solved with a direct solver.

To show numerically that our reordering scheme improves the efficiency of the direct solver, the Stokes problem is solved with a direct solver with various renumbering and reordering combinations shown in Table 5.3. With p-last reordering, - with various renumbering schemes - we do not see much difference in CPU time consumed by the direct solver to get the exact solution. However, using p-last per level, the efficiency of the direct solver increases enormously. Sloan renumbering with p-last per level reordering gives better results than the other combinations. Though a better choice of a renumbering scheme also enhances the efficiency of the direct solver, we see that the increase is largely due to the p-last per level reordering strategy.

Table 5.2: Profile and bandwidth reduction in the backward facing step with Q2-Q1 discretization.

Grid	Profile $\frac{p-last\ per\ level}{p-last}$		Bandwidth $\frac{p-last\ per\ level}{p-last}$	
-	Sloan	CMK	Sloan	CMK
4×12	0.37	0.61	0.18	0.17
8×24	0.28	0.54	0.13	0.08
16×48	0.26	0.50	0.11	0.04
32×96	0.25	0.48	0.06	0.02

Table 5.3: The Stokes backward facing step solved with a direct solver with Q2-Q1 discretization.

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

5.2.2 Properties of the saddle point ILU solver (SILU)

We have tested and compared the SILU preconditioner for the Stokes and Navier-Stokes problems in the domains mentioned in Test Case 1 and Test Case 2. In Table 5.4, Sloan renumbering for grid points is used to solve the Stokes problem in the 2D square domain for the Q2-Q1 discretization. The term "Iter." gives the number of outer iterations for Krylov methods. For small problems, a direct method gives the solution faster than iterative methods coupled with the SILU preconditioner. However, memory requirements for the direct method are large compared to the iterative solution methods [68] and beyond a certain number of grid points the time required to solve the Stokes equations by a direct method increases considerably. The Krylov solvers converge faster for p-last per level than the p-last reordering of the unknowns. The time taken by the convergence of Bi-CGSTAB is less than GMRES(20). Also Bi-CGSTAB uses less matrix-vector products than GMRES(20). The same convergence behavior has been found for the Q2-P1 discretization.

Table 5.4: Solution of the Stokes problem with the Q2-Q1 discretization in the square domain with an *accuracy* of 10^{-6} (Time = total time).

Solver.	Renumber	16×16		32×32		64×64	
		Iter.	Time(s)	Iter.	Time(s)	Iter.	Time(s)
Direct	p-last	-	0.61	-	20.34	-	1378
	p-last per level	-	0.13	-	2.28	-	37
GMRES(20)	p-last	95	0.16	354	1.72	1800	44.0
	p-last per level	50	0.12	207	1.14	792	20.0
Bi-CGSTAB	p-last	36	0.11	90	0.92	255	11.98
	p-last per level	25	0.09	59	0.66	135	6.74

With the p-last per level reordering, the effect of renumbering the mesh by the Sloan or the Cuthill-McKee algorithm is shown in Table 5.5. The Sloan renumbering gives faster convergence than the Cuthill-McKee renumbering for both Q2-Q1 and

Q2-P1 discretizations. The difference in the number of iterations for both renumbering schemes is more pronounced in Q2-P1 discretization. The Sloan renumbering produces a much better profile than Cuthill-McKee renumbering. With a better profile, an incomplete LU decomposition gives a better approximation of the exact LU decomposition and increases the convergence of the preconditioned Krylov subspace method. However, if the problem is highly nonsymmetric and far from being diagonally dominant, the norm of residual $R = A - \hat{L}\hat{U}$ is not a good estimate for the quality of the preconditioner. The Frobenius norm of the term $R(\hat{L}\hat{U})^{-1} = I - A(\hat{L}\hat{U})^{-1}$ gives better insight in the quality of the preconditioner. Even if R is small in norm, it may happen that the preconditioned matrix deviates largely from identity due to very large entries in $(\hat{L}\hat{U})^{-1}$[51], [21].

Table 5.5: Effect of mesh renumbering on convergence of Bi-CGSTAB for various discretizations in the backward facing step Stokes problem with p-last per level and $accuracy = 10^{-6}$.

Grid	Q2-Q1		Q2-P1	
-	Sloan	Cuthill-McKee	Sloan	Cuthill-McKee
-	Iter.	Iter.	Iter.	Iter.
8×24	9	15	29	97
16×48	22	32	40	288
32×96	59	65	73	1300

In case of Navier-Stokes, the number of accumulated inner iterations increases with the increase in Reynolds number and the number of grid points is shown in Figure 5.6. The linear equations in the inner iteration are solved with an accuracy of 10^{-2}. Figure 5.6 shows the accumulated number of inner iterations for solving the linear problems linearized by Picard and Newton. The SILU preconditioned Krylov subspace methods with Sloan renumbering and p-last per level reordering were applied. There are two reasons for the increase in the accumulated inner iterations when the SILU preconditioner is employed:

With the increase in Reynolds number, the number of outer iterations increases, however on average the number of inner iterations remains the same. Therefore, the convergence of the SILU preconditioner itself is hardly influenced by an increase in the Reynolds number. With the increase in the number of grid points, the number of inner iterations increases that gives rise to an increase in the accumulated inner iterations. Change in the number of grid points has less effect on the outer iterations.

In most cases it appears that the Newton method results in faster convergence than the Picard method. However, for a high Reynolds number - due to a bad initial estimate - Newton's method diverges, so it is good practice to start with Stokes equations and then a few Picard iterations followed by Newton iterations. Though the Picard method converges in more outer iterations, the average number of inner iteration required by Picard method is less than for Newton's method. Note that we did not use an upwind technique in our experiments. We expect that convergence will be better in combination with Streamline-Upwind/Petrov-Galerkin (SUPG).

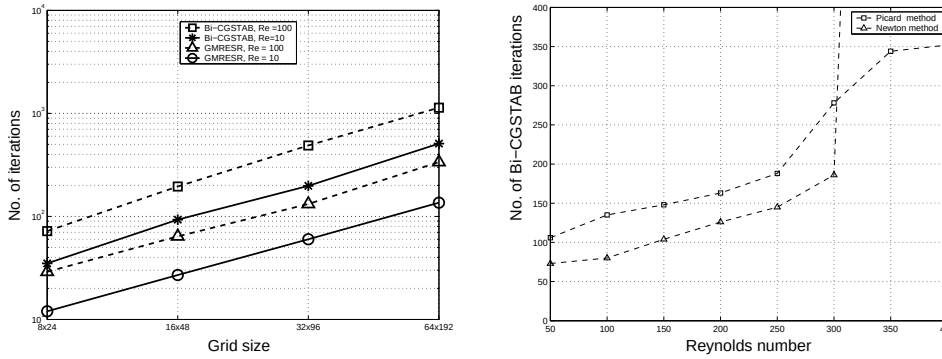


Figure 5.6: Effect of grid increase (left) and Reynolds number (right) on the inner iterations (accumulated) for the Navier-Stokes backward facing step problem with an *accuracy* of 10^{-2} using the p-last per level reordering.

The Stokes and Navier-Stokes equations are also solved for a 3D backward facing step. Results with Q2-Q1 elements for the Stokes problem are shown in Table 5.6. The Cuthill-McKee numbering shows faster convergence than the Sloan renumbering with both p-last and p-last per level reordering schemes. A Stokes problem is also solved with a direct solver in a $8\times8\times8$ grid with p-last per level ordering. It gives the solution in 20 seconds with the Sloan renumbering and 52 seconds with the Cuthill-McKee renumbering. This approach consumes a huge amount of memory. For Q2-P1 elements, the Sloan renumbering gives faster convergence than Cuthill-McKee for both reordering methods. Results given in Table 5.7 suggest that the Krylov subspace method converges in the same number of iterations with both orderings for the Cuthill-McKee renumbering. A possible explanation is that levels in 3D are much larger than in 2D, since in 2D they consist of "lines", whereas in 3D they are essentially "planes". This is the same as the difference in bandwidth between 2D and 3D. Hence effect of levels is much more pronounced in 2D than in 3D. Table 5.8 shows the results obtained from the solution of the Navier-Stokes problem. This reveals that the preconditioned Krylov methods have the same convergence behavior for various Reynolds numbers and grid sizes, as we have observed in the Stokes problem.

Table 5.6: Solution of the 3D Stokes backward facing step problem using Q2-Q1 elements with Bi-CGSTAB and *accuracy* = 10^{-4}.

	Sloan				Cuthill-McKee			
Grid	p-last		p-last per level		p-last		p-last per level	
	Iter.	Time(s)	Iter.	Time(s)	Iter.	Time(s)	Iter.	Time(s)
$8\times8\times12$	64	2.1	62	2	60	1.76	44	1.63
$16\times16\times24$	164	42	140	40	140	27.5	100	23
$24\times24\times36$	274	266	236	251	252	215	188	123

Table 5.7: Solution of the 3D Stokes backward facing step problem using Q2-P1 elements with Bi-CGSTAB and *accuracy* = 10^{-4}.

	Sloan				Cuthill-McKee			
Grid	p-last		p-last per level		p-last		p-last per level	
	Iter.	Time(s)	Iter.	Time(s)	Iter.	Time(s)	Iter.	Time(s)
$8 \times 8 \times 12$	53	3.72	48	3.45	64	4.20	64	4.16
$16 \times 16 \times 24$	138	114	107	92	278	214	231	181
$24 \times 24 \times 36$	295	591	255	512	425	833	426	836

Table 5.8: Accumulated inner iterations for the 3D Navier-Stokes backward facing step problem with p-last per level reordering.

Picard method	Q2-Q1		Q2-P1	
Reynolds number = 75, Bi-CGSTAB, *accuracy* = 10^{-2}				
Grid	Sloan	Cuthill-McKee	Sloan	Cuthill-McKee
$8 \times 8 \times 12$	147	76	253	350
$16 \times 16 \times 24$	575	222	598	934
$24 \times 24 \times 36$	8634	533	1994	2441
Reynolds number = 100				
$8 \times 8 \times 12$	150	85	318	376
$16 \times 16 \times 24$	694	249	690	1432
$24 \times 24 \times 36$	NC	576	1636	2668

The effect of grid stretching is shown in Table 5.9 and 5.10 with all possible combinations of renumberings. We increased the number of elements in one direction only, so that the aspect ratio of the elements increases. Furthermore, some fill-in and lumping is introduced in the SILU preconditioner for a stretched grid. This improves the convergence of the iterative method. However, the CPU time and memory usage increases. So this suggests that fill-in and lumping should only be used when the ILU preconditioned iterative method is diverging. We see a sharp dive (minimum) in the number of iterations for the 64×24 grid. For this behavior we do not yet have an explanation. This has been observed for cells with an 8:3 ratio in the backward facing step. For both reordering schemes the number of iterations increases with the increase in stretching, but there is a clear difference between Sloan and Cuthill-McKee renumbering. Cuthill-McKee sometimes diverges, while Sloan in combination with p-last per level always converges. The convergence is also improved in some cases where the incompressibility constraint is relaxed with a parameter ϵp, see Table 5.11. However, it is difficult to find a suitable value of ϵ. In Figure 5.7, we see that the reduction in number of iterations with the increase in ϵ is not linear for the Q2-P1 discretization in the Stokes problem. Though the number of iterations decreases for higher values of ϵ, there is a large increase in the error norm as well. For the Navier-Stokes problem the decrease in number of iterations is not large as shown in Figure 5.8. With a slight compromise on the error norm, a suitable value of ϵ can be found in the range 10^{-10} to 10^{-6}. We have made the same observation for Q2-Q1 elements.

Table 5.9: Solution of the Stokes problem in a stretched backward facing step with Bi-CGSTAB with p-last ordering.

Q2-P1	Sloan renumbering, *accuracy* = 10^{-4}			
	SILU	SILUF	Lumped	SILUF,Lumped
Grid	p-last iter.-time(s)			
8×24	36(0.04)	14(0.06)	41(0.04)	17(0.06)
16×24	55(0.15)	27(0.2)	71(0.18)	33(0.24)
32×24	364(1.78)	138(1.36)	305(1.5)	89(1.13)
64×24	NC	NC	>3000*1	237(5.31)*2
128×24	NC	NC	NC	780(34)*3
	Cuthill-McKee renumbering, *accuracy* = 10^{-4}			
8×24	158(0.16)	14(0.08)	140(0.15)	17(0.1)
16×24	281(0.64)	26(0.34)	231(0.51)	36(0.41)
32×24	277(1.36)	50(0.96)	520(2.53)	71(1.24)
64×24	>3000	586(18.59)	NC	209(7)*5
128×24	NC	NC	NC	727(47.34)*6

Table 5.10: Solution of the Stokes problem in a stretched backward facing step with Bi-CGSTAB using p-last per level ordering.

Q2-P1	Sloan renumbering, *accuracy* = 10^{-4}			
	SILU	SILUF	Lumped	SILUF,Lumped
Grid	p-last iter.-time(s)			
8×24	23(0.03)	8(0.02)	27(0.04)	8(0.03)
16×24	47(0.12)	16(0.11)	52 (0.14)	19(0.12)
32×24	159(0.8)	60(0.63)	599 (2.91)	55(0.61)
64×24	58(0.65)	18(0.45)	217(2.22)	66 (1.26)
128×24	293(6.1)*4	-	808(16.34)	224(7.6)
	Cuthill-McKee renumbering, *accuracy* = 10^{-4}			
8×24	148(0.15)	7(0.07)	175 (0.17)	9(0.07)
16×24	287(0.65)	20(0.25)	258 (0.58)	21(0.26)
32×24	276(1.31)	45(0.88)	568 (2.71)	50(0.96)
64×24	>3000	14(0.88)	-	51(2.1)
128×24	NC	17(2)	NC	120(9.33)

Table 5.11: Effect of ϵ on the convergence with cases labeled with * in Table 5.9 and 5.10.

case	ϵ	iterations- time(sec)	case	ϵ	iterations- time(sec)
*1	1e-10	229(2.37 sec)	*2	1e-10	150(3.52 sec)
*3	1e-09	405(17.8 sec)	*4	1e-10	261(5.46 sec)
*5	1e-10	137(4.73 sec)	*6	1e-10	313(21.0 sec)

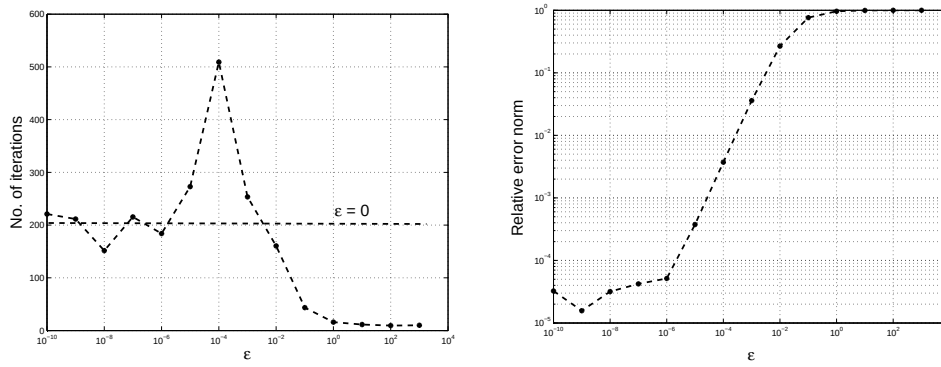


Figure 5.7: Effect of the incompressibility relaxation ϵ on the number of iterations (left) and the relative error norm (right) in the backward facing Stokes problem using Bi-CGSTAB with an *accuracy* of 10^{-4}, and Cuthill-McKee renumbering with p-last per level reordering for the 16×48 Q2-P1 discretization.

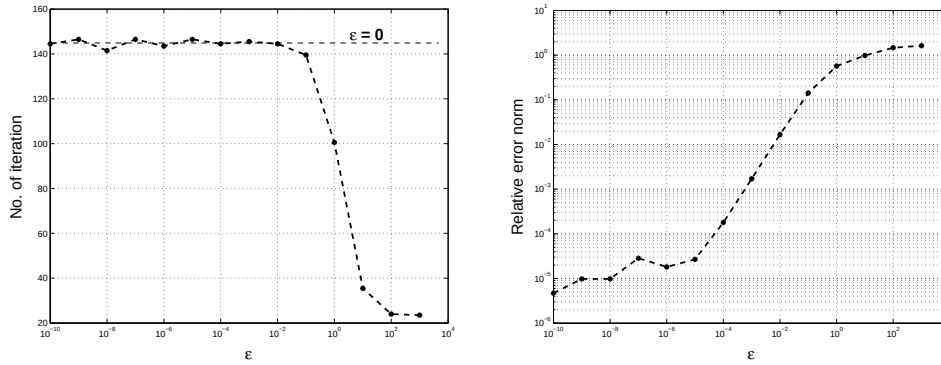


Figure 5.8: Effect of the incompressibility relaxation ϵ on the number of iterations (left) and the relative error norm (right) in the backward facing Navier-Stokes problem using Bi-CGSTAB with an *accuracy* of 10^{-4}, and Cuthill-McKee renumbering with p-last per level reordering for the 16×48 Q2-P1 discretization with $Re = 100$ in the last step of the Picard iteration.

5.3 Summary

In this chapter, we discussed a preconditioner based on ILU factorization of the Navier-Stokes matrix. It is well known that a straightforward application of this method can lead to break down or bad convergence due to small pivot elements. In order to prevent this the grid points are first renumbered with the classical Cuthill-McKee or Sloan method. Thereafter we reorder the unknowns such that on each level the velocity unknowns are ordered before the pressure unknowns. In our experiments we never observed any break down of the SILU decomposition and the convergence is fast for a large range of problems.

We observed the following properties of the SILU preconditioner:

- in 2D problems, Sloan renumbering with p-last per level reordering leads to the best results for Taylor-Hood and Crouzeix-Raviart elements,
- in 3D problems, Cuthill-McKee renumbering gives fast convergence for the Q2-Q1 discretization, whereas for the Q2-P1 discretization, Sloan renumbering gives a better convergence,

The number of iterations increases with the increase in the number of elements in the grid and increases mildly with the increase in the Reynolds number. For this reason we consider block preconditioners in the next chapter which appear to be less sensitive to the grid size and Reynolds number.

Chapter 6

SIMPLE-type preconditioners

In this chapter, we discuss block preconditioners that are based on the SIMPLE method formulation. SIMPLE (semi-implicit pressure linked equation) is used by Patanker as an iterative method to solve the Navier-Stokes problem [60]. The convergence of the iterative method depends on relaxation parameters for the velocity and pressure. The scheme belongs to the class of classical iterative methods and gives slow convergence. Still the scheme is very popular in the CFD community and has been used in many commercial packages, for example FLUENT[1].

A much faster convergence can be achieved if the SIMPLE method is accelerated with a Krylov method. Variants of SIMPLE (SIMPLER, SIMPLEC) are also used as preconditioners to solve the Navier-Stokes problem [32].

Two new SIMPLE variants are proposed here. They are called hSIMPLER (hybrid SIMPLER) and MSIMPLER (Modified SIMPLER). We show that these new variants give very good convergence. Moreover, the effect of relaxation parameters used in SIMPLE is also discussed.

6.1 SIMPLE-type preconditioner

Originally SIMPLE has been developed for finite volume and finite difference discretizations [92], [60]. The algorithm is based on the following steps. First the pressure is assumed to be known from the prior iteration. Then the velocity is solved from the momentum equations. The newly obtained velocities do not satisfy the continuity equation since the pressure is only a guess. In the next substeps the velocities and pressures are corrected in order to satisfy the discrete continuity equation. The Patankar formulation for FVM is written in the form of a distributive iterative method (block matrices) by Wesseling [92].

[1] http://www.fluent.com/

In this chapter we apply SIMPLE-type preconditioners for the incompressible (Navier-) Stokes equations discretized by the finite element method. Before going into details, we first give some definitions based on the block $\mathcal{L}_b\mathcal{D}_b\mathcal{U}_b$ decomposition given in (4.5):

Definition 6.1.1.

$$\mathcal{L}_{bt} = \mathcal{L}_b\mathcal{D}_b = \begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix}, \tag{6.1}$$

and

$$\mathcal{U}_{bt} = \mathcal{D}_b\mathcal{U}_b = \begin{bmatrix} F & B^T \\ 0 & \hat{S} \end{bmatrix}, \tag{6.2}$$

where $\hat{S}$ represents a Schur complement matrix approximation. Later on, we will discuss possible choices for $\hat{S}$.

$$\hat{\mathcal{U}}_b = \begin{bmatrix} I & M_u^{-1}B^T \\ 0 & I \end{bmatrix}, \tag{6.3}$$

$$\hat{\mathcal{L}}_b = \begin{bmatrix} I & 0 \\ BM_l^{-1} & I \end{bmatrix}, \tag{6.4}$$

where M_l and M_u are approximations of F. In case SIMPLE is used as preconditioner we have to solve a system $Pz = r$ in each iteration. We will split this in a velocity and pressure part as solving $Pz = r$, where z and r are already defined in Section 4.3.

6.2 SIMPLE preconditioner

The SIMPLE method as a preconditioner represented in a block matrix form is defined as:

$$P_S = \mathcal{L}_{bt}\hat{\mathcal{U}}_b, \tag{6.5}$$

with $\hat{S} = BD^{-1}B^T$ and $M_u = D$, where D is the diagonal of the velocity matrix. To solve $P_S z = r$ can be done with one iteration of the method given in Algorithm 6.1.

Vuik et al [89], used SIMPLE and its variants as a preconditioner with GCR to solve the incompressible Navier-Stokes equations. One iteration of the SIMPLE algorithm with assumption $p^* = 0$ is used as preconditioner. The preconditioner consists of one velocity solve and one pressure solve.

Proposition 6.2.1. *For the SIMPLE preconditioned matrix $\tilde{A}$,*

- *1 is an eigenvalue with multiplicity n_u, and*
- *the remaining eigenvalues are defined by the generalized eigenvalue problem $S\,p = \lambda\hat{S}\,p$.*

Algorithm 6.1 SIMPLE algorithm

1. p^* is given
2. Solve $Fu^* = r_u - B^T p^*$.
3. Solve $\hat{S}\delta p = r_p - Bu^*$.
4. update $z_u = u^* - D^{-1}B^T\delta p$.
5. update $z_p = p^* + \delta p$.
6. If not converged goto 2

Proof: [45]

The eigenvalues of the preconditioned system can be obtained from the generalized eigenvalue problem:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & 0 \\ B & \hat{S} \end{bmatrix}\begin{bmatrix} I & D^{-1}B^T \\ 0 & I \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix}. \tag{6.6}$$

This can be written as:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix} = \lambda \begin{bmatrix} F & FD^{-1}B^T \\ B & 0 \end{bmatrix}\begin{bmatrix} u \\ p \end{bmatrix}, \tag{6.7}$$

that is

$$\begin{aligned} Fu + B^T p &= \lambda(Fu + FD^{-1}B^T p), \\ Bu &= \lambda Bu. \end{aligned} \tag{6.8}$$

Multiplying the first equation in (6.8) by F^{-1} and re-arranging the two equations gives

$$\begin{aligned} (1-\lambda)u &= (\lambda D^{-1} - F^{-1})B^T p, \\ B(1-\lambda)u &= 0. \end{aligned} \tag{6.9}$$

From (6.9), it can be seen that 1 is an eigenvalue of (6.7). Note that the matrix $(D^{-1} - F^{-1})$ is non-singular by assumption. The eigenvectors corresponding to eigenvalue 1 are $v_i = \binom{u_i}{0} \in \mathbb{R}^{(n_u+n_p)}$, $u_i \in \mathbb{R}^{n_u}$, where $\{u_i\}_{i=1}^{n_u}$ is a basis of $\mathbb{R}^{n_u}$.
If $\lambda \neq 1$ then $Bu = 0$ in (6.9). Multiplying the first equation in (6.9) by B gives:

$$\begin{aligned} 0 &= \lambda BD^{-1}B^T - BF^{-1}B^T p, \\ \lambda BD^{-1}B^T &= BF^{-1}B^T p. \end{aligned} \tag{6.10}$$

If follows that

$$Sp = \lambda \hat{S} p, \text{ where } S = -BF^{-1}B^T \text{ and } \hat{S} = -BD^{-1}B^T. \tag{6.11}$$

Figure 6.1 shows the eigenvalue spectrum of the Navier-Stokes system and the SIMPLE preconditioned system. The rectangular region shown is zoomed in the next subfigure. It shows that the SIMPLE preconditioner improves the overall spectrum and clusters most of the eigenvalues around 1. The eigenvalues equal to 1 are expected equal to the number of velocity unknowns. Remaining eigenvalues depend on the approximation of the Schur complement matrix. More details on eigenvalue analysis are given in [45].

The preconditioner converges nicely if used in combination with the GCR method. However, the convergence rate suffers from an increase in the number of grid elements and Reynolds number.

6.2.1 SIMPLER

A variant of SIMPLE, known as SIMPLER is supposed to give Reynolds independent convergence. Instead of estimating the pressure p^* in the SIMPLE algorithm, p^* is obtained from solving a subsystem:

$$\hat{S}\, p^* = r_p - BD^{-1}((D-F)u^k + r_u). \tag{6.12}$$

where u^k is obtained from the prior iteration. In case SIMPLER is used as preconditioner, u^k is taken equal to zero. The classical SIMPLER algorithm proposed by Patanker consists of two pressure solves and one velocity solve. In the literature the SIMPLER algorithm is formulated such that the steps of the algorithm are closely related to the Symmetric Block Gauss Seidel method [88]. This form of the SIMPLER preconditioner can be written as:

$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \begin{pmatrix} u^k \\ p^k \end{pmatrix} + \mathcal{U}_{bt}^{-1}\, \hat{\mathcal{L}}_b{}^{-1} \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - \mathcal{A} \begin{pmatrix} u^k \\ p^k \end{pmatrix} \right), \tag{6.13}$$

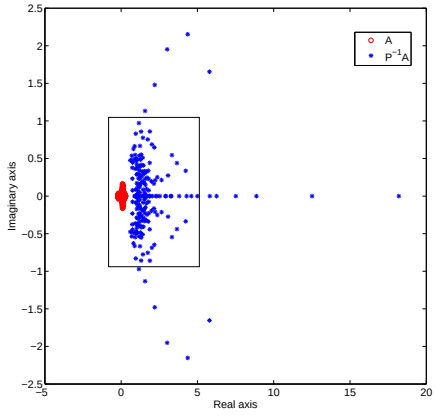
$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + \hat{\mathcal{U}}_b{}^{-1}\, \mathcal{L}_{bt}^{-1} \left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - \mathcal{A} \begin{pmatrix} u^* \\ p^* \end{pmatrix} \right), \tag{6.14}$$

where $\mathcal{A}$ represents the complete matrix given in (2.21), $\hat{S} = -BD^{-1}B^T$, $M_l = D$, $M_u = D$, u^k and p^k in (6.13) are obtained from the previous step (both zero in the preconditioner case). This formulation is quite expensive if used as preconditioner, because the steps given in (6.13) and (6.14) contain two Poisson solves, and two velocity subproblems solves as opposed to one velocity solve in the classical algorithm and matrix-vector updates. However, the extra velocity solve in formulation (6.13) and (6.14) has no significant effect on the convergence with the SIMPLER preconditioner.

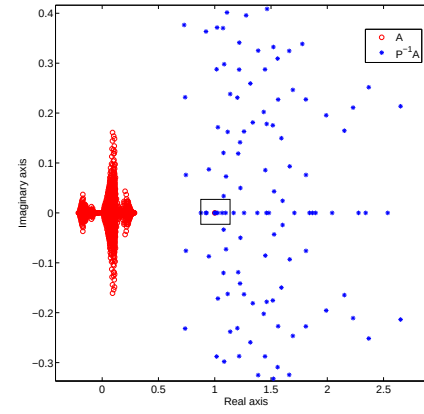
Lemma 6.2.1. *In the SIMPLER preconditioner/algorithm, both variants (one or two velocity solves) are identical.*

Proof:
We first start with the choice u^k and p^k are zero vectors. To solve the system $Pz = r$, (6.13) reduces to

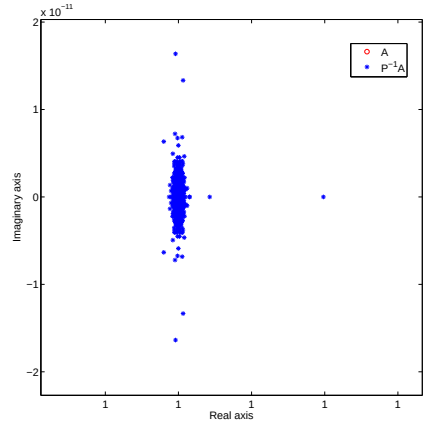
$$\begin{pmatrix} u^* \\ p^* \end{pmatrix} = \mathcal{U}_{bt}^{-1}\, \hat{\mathcal{L}}_b{}^{-1} \begin{pmatrix} r_u \\ r_p \end{pmatrix}. \tag{6.15}$$



(a) Complete picture



(b) Zooming Figure (a)



(c) Zooming Figure (b)

Figure 6.1: Eigenvalues of the Navier Stokes system (at 2nd Picard iteration) (A) and preconditioned with SIMPLE ($P^{-1}A$). 8×24 Q2-Q1 Backward facing step problem with $Re = 100$.

Rewriting (6.15) leads to

$$p^* = \hat{S}^{-1}(r_p - BD^{-1}r_u), \tag{6.16}$$

and

$$u^* = F^{-1}(r_u - B^T p^*). \tag{6.17}$$

Next we consider (6.14). First compute the residual part,

$$\begin{pmatrix} r_{un} \\ r_{pn} \end{pmatrix} = \begin{pmatrix} r_u \\ r_p \end{pmatrix} - \mathcal{A}\begin{pmatrix} u^* \\ p^* \end{pmatrix}. \tag{6.18}$$

The velocity part becomes

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

If we substitute u^* from (6.17) into r_{un} we get

$$r_{un} = r_u - FF^{-1}(r_u - B^T p^*) - B^T p^* = 0, \text{ and the pressure part}$$

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

therefore, (6.14) reduces to

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^* \\ p^* \end{pmatrix} + \hat{\mathcal{U}}_b^{-1}\mathcal{L}_{bt}^{-1}\begin{pmatrix} 0 \\ r_{pn} \end{pmatrix}. \tag{6.19}$$

The formulation (6.15) and (6.19) gives rise to three steps in the SIMPLER preconditioner because there is no need to perform an extra velocity solve in (6.19) when the right-hand side is zero.

$$\delta p = \hat{S}^{-1}(r_p - Bu^*), \tag{6.20}$$

$$u^{k+1} = u^* - D^{-1}B^T\delta p, \tag{6.21}$$

and

$$p^{k+1} = p^* + \delta p. \tag{6.22}$$

This can also be proven for the SIMPLER algorithm with nonzero u^k and p^k, which proves the theorem □

We observe in our numerical experiments that both variants are different if inexact solves are used but the convergence of both variants is nearly the same. SIMPLER is more expensive than SIMPLE. One iteration of the SIMPLER algorithm is approximately 1.3 times more expensive than the SIMPLE iteration [89]. SIMPLER convergence is also faster than the SIMPLE preconditioner. However, both preconditioners give h-dependent convergence.

6.3 Effect of relaxation parameter

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

In our case we use SIMPLE as preconditioner, which means that we apply only one SIMPLE iteration per GCR step. Due to application of one SIMPLE iteration, it is assumed that the introduction of a relaxation parameter will have almost no effect on the convergence of the preconditioned Krylov method because SIMPLE is applied to a problem with different right-hand side at each iteration. We have observed that in some cases introduction of a relaxation parameter ω in the pressure part improves convergence of the SIMPLE preconditioner. Therefore, the last step in the SIMPLE-type preconditioners will be replaced by

$$p = p^* + \omega \delta p. \tag{6.23}$$

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

6.4 Improvements in the SIMPLER preconditioner

In this section, two SIMPLER variants are discussed that improve the convergence of the SIMPLER preconditioner.

6.4.1 hSIMPLER

We have observed that in the Stokes problem, the SIMPLER preconditioner shows stagnation at the start of the iterative method. This behavior is not seen in the SIMPLE preconditioner. This is shown in Figure 6.2. A better convergence can be achieved if the first iteration is carried out with the SIMPLE preconditioner and after that SIMPLER is employed. We call this combination hSIMPLER (hybrid SIMPLER). This implementation gives a fair reduction in the number of iterations if the Stokes problem is solved. However, in the Navier-Stokes problem, SIMPLER performs better than hSIMPLER. More details are given in the section with numerical experiments.

6.4.2 MSIMPLER

Elman et al [35], [31] discussed relation between SIMPLE and commutator preconditioners. The more general form of (4.18) is given by:

$$(BF^{-1}B^T)^{-1} \approx -F_p(BM_1^{-1}B^T)^{-1}, \tag{6.24}$$

where

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

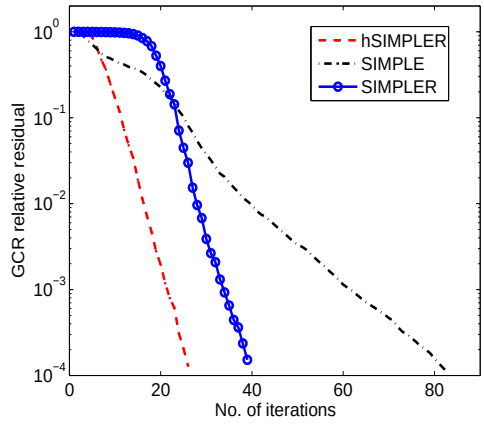


Figure 6.2: Convergence plot of SIMPLE-type peconditioners for the Stokes problem

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

$$P = \mathcal{L}_{bt}\hat{\mathcal{U}}_b \begin{bmatrix} I & 0 \\ 0 & F_p^{-1} \end{bmatrix}. \tag{6.25}$$

When $\hat{S} = -BD^{-1}B^T$, $M_u = D$ and F_p is the identity matrix, then the preconditioner formulation (6.25) corresponds to SIMPLE. The formulation given in (6.25) is equivalent to the SIMPLE algorithm if the subsystem for the pressure part in Step 3 in Algorithm 6.1 is solved with the approximation given in (6.24)

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

When FD^{-1} is close to identity, F_p will also be close to identity. This is true in a time dependent problem with small time steps where the diagonal of F has significantly larger entries than the off-diagonal entries [31].

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

$$F_t = \frac{1}{\Delta t}Q_v + F, \tag{6.26}$$

where F_t represents the velocity matrix for the time dependent problem and Δt represents the time step. For a small time step $F_t \approx \frac{1}{\Delta t}Q_v$. This kind of approximation has been used in fractional step methods for solving the unsteady Navier-Stokes problem [61],[13], [20]. We use this idea in solving the steady Navier-Stokes problem.

Therefore, we choose $M_1 = M_2 = \hat{Q}_v$ in (6.24) resulting in:

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

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

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

and the approximation (6.24) becomes

$$BF^{-1}B^T \approx -B\hat{Q}_v{}^{-1}B^T. \tag{6.27}$$

Based on this result we replace D^{-1} in the SIMPLER algorithm by $\hat{Q}_v{}^{-1}$. We refer to this method as MSIMPLER (Modified SIMPLER) (P_{MSR}) .

Algorithm 6.2 MSIMPLER preconditioner

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

It is clear from Algorithm 6.2 that the cost of the MSIMPLER preconditioner is equal to the cost of the SIMPLER preconditioner. However, in solving the Navier-Stokes problem, at each nonlinear iteration, the Schur complement approximation in the MSIMPLER does not need to be build again because the operators used in the Schur complement approximation are independent of the any change that take place at each nonlinear iteration.

6.4.3 Suitable norm to terminate the Stokes iterations

We have noticed that when we solve the Stokes equations by SIMPLE-type preconditioned GCR method, the number of iterations depends on the viscosity (Reynolds number). See for example Table 6.1. Such a result is unexpected since in the Stokes equations the viscosity is only a scaling parameter and therefore the convergence should be independent of the Reynolds number. Close inspection reveals that the accuracy of the solution is lower for high Reynolds numbers than for low ones. Hence the conclusion is that, in case of Stokes, the termination criterion should be adapted to avoid this viscosity dependence.

Table 6.1: Backward facing step: Solution of the Stokes problem with SIMPLER preconditioned GCR (*accuracy* of 10^{-6}).

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

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

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} f \\ g \end{bmatrix}. \tag{6.28}$$

The convergence criterion for the outer iterations is:

$$\frac{\left\| \begin{bmatrix} f - Fu - B^T p \\ g - Bu \end{bmatrix} \right\|_2}{\left\| \begin{bmatrix} f \\ g \end{bmatrix} \right\|_2} \leq \epsilon. \tag{6.29}$$

Since $\nu = 1$, the solution is obtained up to the required accuracy because no scaling is involved in the momentum and continuity equations and convergence check (6.29) will terminate the iterative method at the desired accuracy. In case of a general value of the ν we rewrite the system as:

$$\begin{bmatrix} \tilde{F} & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ \tilde{p} \end{bmatrix} = \begin{bmatrix} \tilde{f} \\ g \end{bmatrix}, \tag{6.30}$$

where $\tilde{F} = \nu F$, $\tilde{p} = \nu p$ and $\tilde{f} = \nu f$. The SIMPLE preconditioner for (6.30) can be written as:

Effect of ν on the SIMPLE preconditioner

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

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

- For the outer iterations the usual termination criterion is

$$\frac{\left\|\begin{bmatrix} \tilde{f} - \tilde{F}u - B^T\tilde{p} \\ g - Bu \end{bmatrix}\right\|_2}{\left\|\begin{bmatrix} \tilde{f} \\ g \end{bmatrix}\right\|_2} \leq \epsilon. \tag{6.31}$$

We see in (6.30), that only the momentum equation is scaled with ν so this will effect the convergence of the outer iterative method if convergence check (6.31) is applied. This means that using ν in the Stokes problem effects the accuracy of the solution of the Stokes problem. If a suitable norm is used, ν will have no effect on the convergence of the iterative method. First we shall define some quantities:

$$N_{full} = \left\|\begin{bmatrix} \tilde{f} - \tilde{F}u - B^T\tilde{p} \\ g - Bu \end{bmatrix}\right\|_2, \quad N_r = \left\|\begin{bmatrix} \tilde{f} \\ g \end{bmatrix}\right\|_2 \tag{6.32}$$

$$N_u = \left\| \tilde{f} - \tilde{F}u - B^T\tilde{p} \right\|_2, \quad N_{ru} = \left\| \tilde{f} \right\|_2 \tag{6.33}$$

$$N_p = \left\| g - Bu \right\|_2, \quad N_{rp} = \left\| g \right\|_2 \tag{6.34}$$

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

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

2. $N_u \leq \epsilon N_{ru}$ *and* $N_p \leq \epsilon N_{rp}$
   Fails to show convergence in the Navier-Stokes problem due to too small ϵN_{rp}. However, in the Stokes problem it shows viscosity independent convergence. Moreover, there is a chance that N_{rp} is zero due to certain boundary conditions.

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

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

In our implementation, we tested all these conditions and the third option seems to be the best condition both in the Stokes and the Navier-Stokes problem since this is the only condition that encounters the effect of scaling on convergence without any problem.
In the next section, we perform some numerical experiments to test convergence of our preconditioners.

6.5 Numerical Experiments

The preconditioners discussed in this chapter are employed to solve Test Cases 1 and 2. To solve the subsystems iteratively, MG and ILU preconditioned Krylov subspace methods are used. The iteration is stopped if the linear systems satisfy $\frac{\|r^k\|_2}{\|b\|_2} \leq tol$, where r^k is the residual at the kth step of the Krylov subspace method, b is the right-hand side, and tol is the desired tolerance value. Some abbreviations used are: t_s for time in seconds, *Iter.* =total iterations for the nonlinear problem, NC for no convergence, in-it-u and in-it-p for the number of iterations taken by the solver to solve subsystems in the preconditioners corresponding to the velocity and pressure part, respectively. The SEPRAN [2](written in FORTRAN) and IFISS packages [3](written in MATLAB) are used to solve the problems. Numerical experiments are performed on an *Intel 2.66 GHz processor with 8GB RAM*.

As already reported in [89], the convergence with SIMPLE preconditioner depends on the mesh size and Reynolds number. Before discussing the SIMPLE-type preconditioner comparison in Section 6.5.2, we investigate the effect of relaxation parameter on the convergence of the SIMPLE preconditioner.

6.5.1 Effect of relaxation parameter

Experiments revealed that application of a relaxation parameter for the velocity part gives no improvement in the convergence. Therefore, we restrict ourselves to varying the relaxation parameter ω in (6.23). Since relaxation did not improve the convergence of SIMPLER only the SIMPLE preconditioner is considered.

In the case of Stokes, choosing ω properly, reduces the number of iterations with a factor 3 or 4. For example, in Figure 6.3, the optimal value of ω (0.05) reduces the number of outer iterations from 193 to 59 for a 64×64 grid. The reduction in the number of inner iterations - not shown in figure - is from 1400 to 77 for the velocity subsystem and 3600 to 1200 for the pressure subsystem. In Figure 6.4, it can be seen that a suitable relaxation parameter gives nice convergence and reduction in CPU time in solving the backward facing step Stokes problem for various grid sizes.

Table 6.2 shows the effect of ω for various values of Reynolds number (Re) in the Navier-Stokes equations. We can see from the table that a proper value of ω may give some gain, but the profit is only small compared to that for the Stokes problem. Furthermore it is clear that the optimal value of ω is different for Stokes and Navier-Stokes. The probable cause is that the pressure approximation in the second step of the nonlinear iteration is much better than in the first iteration. A good value of ω is problem dependent and therefore not pleasant to use.

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

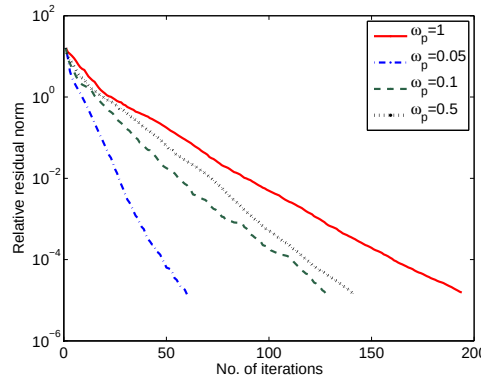
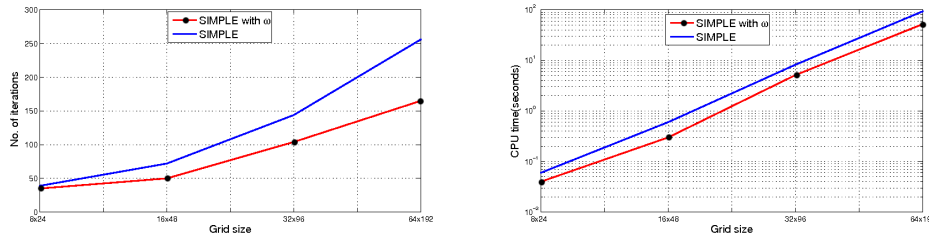


Figure 6.3: The Stokes problem solved with 64×64 Q2-Q1 elements discretized driven cavity problem with varying ω.



(a) Effect on convergence

(b) Effect on CPU time

Figure 6.4: Effect of ω on convergence of the SIMPLE preconditioner solving the Stokes backward facing step problem with increase in grid size.

Table 6.2: Effect of relaxation on the Navier-Stokes problem with a solution accuracy 10^{-6}.

Re	$\omega = 1$	$\omega = 0.5$	$\omega = 0.35$	$\omega = 0.3$	$\omega = 0.2$
	Iter.	Iter.	Iter.	Iter.	Iter.
100	657	641	552	552	563
200	870	803	773	857	783
500	7637	7080	6800	6666	NC

6.5.2 Comparison of SIMPE-type preconditioners

SIMPLE-type preconditioners are employed to solve the 2D backward facing step Stokes problem. These preconditioners are combined with the GCR method ([30]), since this method allows variable preconditioners.

Table 6.3 shows the computation time and number of iterations for a series of grids. For each grid, experiments are performed twice, first with low inner accuracy (upper row) and then with high inner accuracy (lower row).

In case of SIMPLE and MSIMPLER a low inner accuracy implies that the subsystems for the velocities are solved with a relative accuracy of 10^{-1}, and the systems for the pressure with an accuracy of 10^{-2}. In case of high accuracy we used 10^{-6} both for the velocity and pressure. For SIMPLER and hSIMPLER it was necessary to increase the accuracy for the inner solves for increasing grid size. Otherwise no convergence could be reached. The motivation to compare low and high inner accuracies is to investigate the dependence of the SIMPLE-type preconditioners on the inner accuracy. From Table 6.3 it is clear that MSIMPLER is the best choice both with respect to the number of iterations as to the CPU time. Increasing the inner accuracy has only a small effect on the number of GCR iterations but a considerable negative effect on the CPU time.

Figure 6.2 shows that SIMPLER stagnates in the start of iterations. This behavior has been erased by using hSIMPLER. The necessary increase of inner accuracies for finer grids for SIMPLER and hSIMPLER is visible in the smaller difference between upper and lower row in Table 6.3. We also see that SIMPLER does not converge at all for fine grids. The behavior of these preconditioners for a Navier-Stokes flow (driven cavity) is shown in Figure 6.5. A fixed 64×64 grid with Q2-Q1 elements are used.

Table 6.3: Stokes backward facing step solved with preconditioned GCR(20) with *accuracy* of 10^{-6} , PCG used as an inner solver.

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

The Reynolds number is varied from 100 to 1000 and no upwinding is applied. In all cases the low inner accuracy of the upper row in Table 6.3 is used. The left-hand figure shows the average number of inner iterations per Picard step and the right-hand figure the overall CPU time, which increases due to the increase of Picard iterations when Reynolds increases. We see that the average number of inner iterations per Picard step depends mildly on the Reynolds number. Again MSIMPLER proves to be superior to the other SIMPLE-type preconditioners.

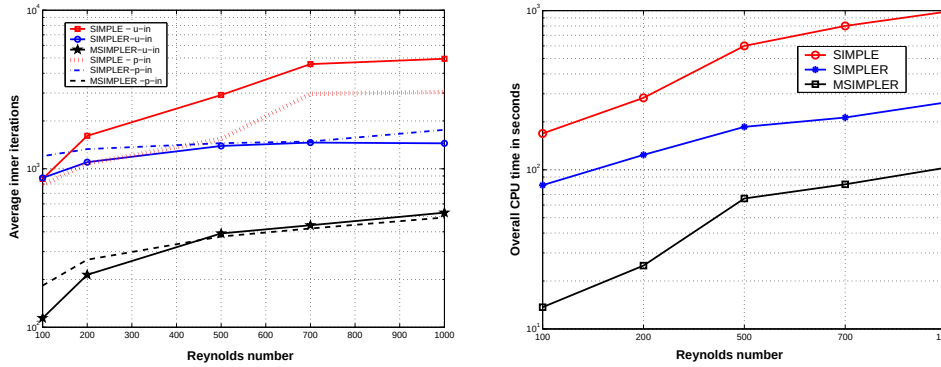


Figure 6.5: The Navier-Stokes problem solved with 64×64 Q2-Q1 elements discretized driven cavity problem with varying Reynolds number, Number of average inner iterations (Left), CPU time in seconds (Right)-(SEPRAN)

The eigenvalue spectrum of the system preconditioned with SIMPLER and MSIMPLER is shown in Figure 6.6. We observe that the MSIMPLER preconditioner leads to a much better clustered spectrum than SIMPLER. With SIMPLER, the zoomed region shows that, still most of the eigenvalues are negative or close to zero. This slows down the convergence of the Krylov method.

We apply MSIMPLER to approximate the solution of the 2D backward facing step for various grid sizes and Reynolds numbers. Table 6.4 shows the number of iterations and CPU time in the second Picard step for MSIMPLER. The system of equations for pressure and velocity is solved by one MG cycle. Due to the constant preconditioner, Bi-CGSTAB and IDR(s) can be applied. Although we see that the convergence depends on the Reynolds number for coarse grids this is no longer the case for the finest grid. Furthermore in some cases the number of iterations decreases for fixed Reynolds numbers for finer grids. Presumably this is due to the decrease in cell Reynolds number. The relative good result of the last number in the MSIMPLER column must be because of the better cell Reynolds number.

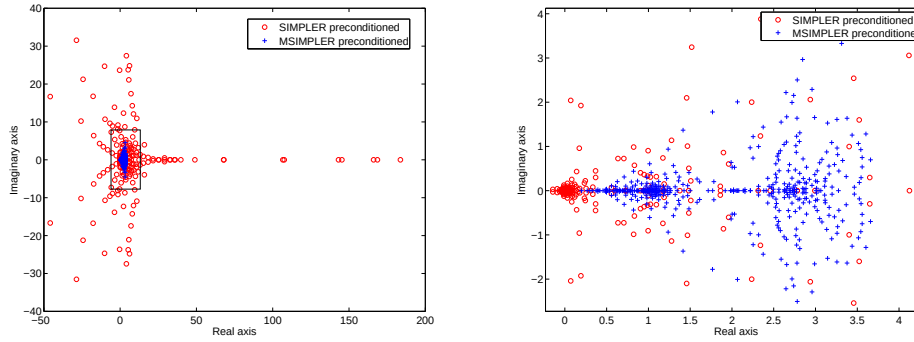
In order to see if the MSIMPLER preconditioner is sensitive to inner accuracies we compare one MG cycle for the inner solver with an exact inner solver in Table 6.5. From this table it is clear that MSIMPLER is hardly effected by the inner accuracy, which is also one of the main advantages of MSIMPLER.

Table 6.4: Backward facing step Navier-Stokes problem (after 2nd Picard iteration) with MSIMPLER preconditioned Bi-CGSTAB with accuracy 10^{-6}. The MG solver is used to solve subsystems (IFISS).

Grid	Re=100	Re=200	Re=400
	Iter. (t_s)		
16×48	9(4.5)	15(7)	29(16)
32×96	11(13.7)	10(17)	15(21)
64×192	20(99)	15(84)	18(102)

Table 6.5: Driven cavity flow problem: The Navier-Stokes problem (after 2nd Picard iteration) is solved with MSIMPLER preconditioned preconditioned Bi-CGSTAB with accuracy 10^{-6}. MG and direct solver is used to solve subsystems (IFISS).

Grid	Re=100	Re=500	Re=1000
	MG/Exact	MG/Exact	MG/Exact
No. of iterations per Picard step			
16×16	10/9	28/25	50/53
32×32	13/12	26/20	45/43
64×64	19/19	25/25	34/32
128×128	25/28	42/44	43/43



(a) Complete picture

(b) Zooming Figure (a)

Figure 6.6: Eigenvalue distribution of the Navier Stokes system (A) and preconditioned with (M)SIMPLER ($P^{-1}A$). 8×24 Q2-Q1 elements discretized Backward facing step problem with $Re = 100$.

6.6 Summary

In this chapter, we discussed SIMPLE-type block preconditioners. SIMPLE together with 3 variants (SIMPLER, hSIMPLER and MSIMPLER) are discussed. SIMPLE(R) as preconditioner is already known in the literature. hSIMPLER performs in the first iteration, SIMPLE and SIMPLER are used for next iterations. SIMPLE and (h)SIMPLER use the diagonal of the velocity matrix in the Schur complement approximation and updates. In MSIMPLER, the diagonal of the velocity mass matrix is used. The outcome of this chapter is:

- The performance of SIMPLER in solving the Stokes problem can be enhanced by employing the first iteration with SIMPLE and then use SIMPLER (hSIMPLER).
- MSIMPLER is at present the fastest of all SIMPLE-type preconditioners.
- In contrast with SIMPLER, MSIMPLER is not sensitive to the accuracies that are used for the inner solvers.
- MSIMPLER is the cheapest to construct of all SIMPLE-type methods since the Schur complement matrix is constant during the nonlinear steps and can therefore be made at the start of the process and reused in the following nonlinear iterations. This is because the scaling is independent of the velocity.
- The number of outer iterations in MSIMPLER hardly increases if a direct solver for the subsystems is replaced by an iterative solver.

In next chapter, we compare SILU, LSC and SIMPLE-type preconditoners for the Stokes and the Navier-Stokes problem.

Chapter 7

Comparison of preconditioners for the incompressible Navier-Stokes equations

In this chapter, we compare the most favourable preconditioners for the incompressible Navier-Stokes equations that are discussed in the previous chapters. Some experiments are also done to compare convergence of IDR(s) and Bi-CGSTAB(ℓ) preconditioned with SILU.

7.1 Preconditioners to be compared

Based on the experiments in previous chapters, we compare convergence of the following preconditioners:

1. SILU (Saddle point ILU preconditioner) (P_{SILU})
2. SIMPLE (P_S)
3. MSIMPLER (P_{MSR})
4. LSC (P_{LSC})

These preconditioners are compared in 2D and 3D using various solvers for the subsystems.

7.1.1 Cost comparison

We define nnz_B as the number of nonzero entries in B, nnz_{Q_p} is the number of nonzero entries in the pressure mass matrix and nnz_F as the number of nonzero entries in F.

We assume that solving the subsystem corresponding to the pressure takes sp flops and the subsystem corresponding to the velocity part takes fu flops. Then the cost of each preconditioner can be written as:

$$Cost_{P_{SILU}} = 2nnz_B + nnz_F + nnz_{Q_p}$$

$$Cost_{P_S} = 4nnz_B + 3n_u + n_p + sp + fu$$

$$Cost_{P_{MSR}} = 8nnz_B + 5n_u + 2n_p + 2sp + fu,$$

and the cost of the LSC preconditioner is

$$Cost_{P_{LSC}} = 6nnz_B + 2nnz_F + 3n_u + 2sp + fu.$$

Per iteration, SILU is cheaper than the other preconditioners. SIMPLE is cheaper per iteration than MSIMPLER since it consists of one velocity and one pressure solve. Computationally, the MSIMPLER preconditioner is less expensive per iteration than the LSC preconditioner. Both preconditioners have to solve three subsystems (2 for the pressure and 1 for the velocity) per iteration. We assume that the cost of solving the subsystem in both preconditioners is the same. Then the difference in cost of both preconditioners consists of the number of matrix-vector multiplications and updates. Per iteration, the difference in cost is:

$$diff = (2nnz_F) - (2n_p + 2n_u + 2nnz_B).$$

If $diff > 0$, MSIMPLER is cheaper than LSC and this appears to be true for all finite elements that satisfy the LBB condition.

7.1.2 Properties of LSC and MSIMPLER

As discussed in Chapter 6, (M)SIMPLER can be written in the form of a symmetric block Gauss Seidel iteration. This can be done by combining (6.13) and (6.14) leading to:

$$\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} u^{k} \\ p^{k} \end{pmatrix} + P_{MSR}^{-1}\left(\begin{pmatrix} r_u \\ r_p \end{pmatrix} - \mathcal{A}\begin{pmatrix} u^{k} \\ p^{k} \end{pmatrix}\right), \tag{7.1}$$

where P_{MSR}^{-1} is the MSIMPLER preconditioner which can be written as

$$P_{MSR}^{-1} = \hat{\mathcal{U}_b}^{-1}\mathcal{L}_{bt}^{-1}\hat{\mathcal{L}_b}^{-1}\mathcal{T}\hat{\mathcal{U}_b}^{-1}\mathcal{U}_{bt}^{-1}\hat{\mathcal{L}_b}^{-1}$$

where

$$\mathcal{T} = \begin{bmatrix} F & 0 \\ 0 & 2\hat{S} \end{bmatrix}.$$

This block Gauss Seidel form of the MSIMPLER preconditioner is based on the complete system [88]. However, in case of an LSC preconditioner, the preconditioner

itself is of block triangular form given in (4.6) and the Schur complement can be seen as a form of the symmetric block Gauss Seidel given by:

$$\hat{S}^{-1} = (BQ_v^{-1}B^T)^{-1}(BQ_v^{-1}FQ_v^{-1}B^T)(BQ_v^{-1}B^T)^{-1}. \tag{7.2}$$

Some common properties of the two preconditioners are:

1. Both preconditioners use the velocity mass matrix in the approximation of the Schur complement matrix.

2. The action of the preconditioner consists of two Poisson solves and one velocity solve.

3. Both show mild dependence on Reynolds number and grid size.

4. Both preconditioners are built from available matrices.

5. The Schur complement may be constructed once -at the start of the linearization because $\hat{Q}_v^{-1}$ remains the same during the linearization steps.

In case of quadrilaterals and hexahedrons $\hat{Q}_v^{-1}$ is the lumped velocity mass matrix, which can also be constructed by using a Newton-Cotes integration rule. In case of quadratic triangles this matrix is singular and we replace it by the diagonal of the consistent mass matrix. For quadratic tetrahedra the situation is somewhat surprising. The lumped velocity mass matrix contains negative entries for the elements corresponding to vertices. Nevertheless the diagonal elements of $B\hat{Q}_v^{-1}B^T$ are all positive. The convergence of MSIMPLER applied to tetrahedra appears to be comparable to that of hexahedra. If we replace $\hat{Q}_v^{-1}$ by the diagonal of the consistent mass matrix, with positive elements only, the convergence of MSIMPLER becomes much slower. We have no explanation for this phenomenon. The same results are also valid for LSC.

7.2 Numerical experiments

In this section, we compare SIMPLE (P_S), MSIMPLER (P_{MSR}) with LSC (P_{LSC}) and SILU. First we consider Test Case 1: the 2D lid driven cavity and Test Case 2: the 2D backward facing step with Q2-Q1 rectangular elements. Next we investigate the behavior for 3D problems using both hexahedra and tetrahedra for the same problems. Finally we test the methods for 2D stretched grids. The first test we apply is the solution of the 2D Stokes backward facing step problem with various grid sizes. SILU performs better with Bi-CGSTAB. With block preconditioners, we use GCR, because it can be used in combination with variable preconditioner. In the last section, we compare IDR(s) and Bi-CGSTAB(ℓ).

7.2.1 Comparison in 2D

In Figure 7.1, preconditioners are compared based on number of outer iterations, CPU time and inner iterations. In terms of all these parameters, we can see that the MSIMPLER performance is better than the other preconditioners. In terms of CPU time, SILU performance is comparable with MSIMPLER and better than other preconditioners. We can see that the increase in the number of iterations with SILU (also SIMPLE) is larger than MSIMPLER (and LSC) for increasing grid size. SIMPLE is robust but expensive, therefore we do not report it in further experiments in 2D.

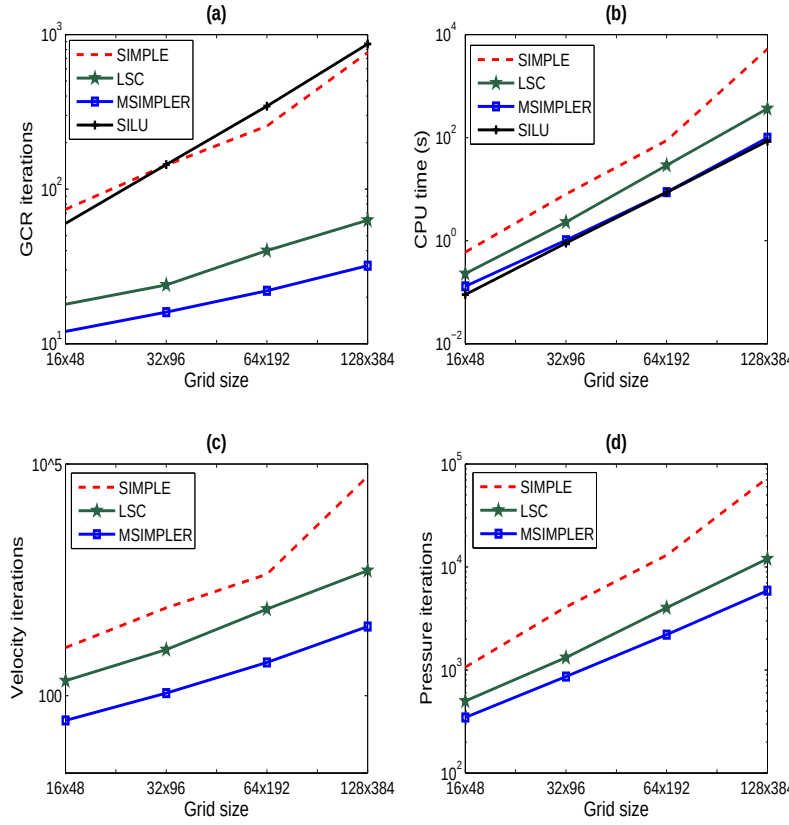


Figure 7.1: 2D Backward facing step (Q2-Q1): The Stokes problem is solved with accuracy 10^{-6}. PCG is used as inner solver in the block preconditioners (SEPRAN)

In Table 7.1 we test block preconditioners where the system of equations for pressure and velocity is solved by one MG cycle. Although we see that for coarse grids the convergence depends on the Reynolds number, this is no longer the case for the finest

Table 7.1: Backward facing step Navier-Stokes problem solved with preconditioned Bi-CGSTAB with accuracy 10^{-6}. MG solver is used to solve subsystems (IFISS).

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

Table 7.2: Backward facing step: Preconditioned GCR is used to solve the Navier-Stokes problem with accuracy 10^{-2}, using Bi-CGSTAB as inner solver, the number of iterations are the accumulated iterations consumed by the outer and inner solvers (SEPRAN)

Grid	P_{LSC}	P_{MSR}	P_{SILU}(Bi-CGSTAB)
	Iter. (t_s)		Iter. (t_s)
Re=100 (11 Picard iterations)			
16×48	114(1.7)	73(1)	246(0.8)
32×96	193(22)	106(10.5)	731(8.7)
64×192	328(545)	182(162)	2071(95)
128×384	695(8863)	296(2806)	6352(1155)
Re=200 (17 Picard iterations)			
16×48	179(2.3)	137(1.7)	436(1.3)
32×96	302(31)	161(14)	1100(13)
64×192	598(983)	232(191)	3114(141)
128×384	946(10405)	541(6301)	2668(9038)
Re=400 (31 Picard iterations)			
16×48	441(4.93)	356(3.9)	716(2.13)
32×96	528(51)	328(25)	1706(20.7)
64×192	NC	405(408)	5366(246)
128×384	NC	663(7025)	NC

grid. Furthermore it is clear that the number of iterations decreases for fixed Reynolds number for finer grids. Presumably this is due to the decrease in cell Reynolds number (*element size*/ν). In all cases MSIMPLER requires less iterations than LSC but the difference becomes small for increasing mesh size. The relative good result of the

last number in the MSIMPLER column must be because of the better cell Reynolds number.

Table 7.2 shows the convergence of MSIMPLER, LSC and SILU where an ILU preconditioned Bi-CGSTAB solver is used for solving subsystems in the block preconditioners. The accuracy for the inner solves is 10^{-2}, which is sufficient to reach the final accuracy of the Navier-Stokes problem without increasing the number of Picard iterations. In this table we report the sum of the iterations in all Picard steps, which gives a complete picture of the whole problem. In this case the difference between MSIMPLER and LSC is much more pronounced. The reason must be the change of inner solver. Furthermore we see that SILU is faster than MSIMPLER except for the finest grid in combination with a large Reynolds numbers.

Table 7.3: Driven cavity flow problem: The Navier-Stokes problem is solved with preconditioned Bi-CGSTAB with accuracy 10^{-6}. MG and direct solver are used to solve subsystems (IFISS).

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

In order to see if the block preconditioners are sensitive to the inner accuracies we compare one MG cycle for the inner solver with an exact inner solver in Table 7.3. From this table it is clear that MSIMPLER is hardly effected by the inner accuracy, whereas LSC is more sensitive in case of coarse grids in combination with a large Reynolds number.

7.2.2 Comparisons in 3D

Iterative solvers for the Navier-Stokes are especially important for 3D problems. In our experiments, we used both hexahedra and tetrahedra. Taylor-Hood elements have been applied due to their availablity in SEPRAN package.

Figure 7.2 shows results of the various preconditioners for the Stokes problem solved on a 3D backward facing step with hexahedral elements. An IC preconditioned CG solver is used as inner solver for the block preconditioners. MSIMPLER requires the least number of iterations (inner/outer) and shows almost grid independent convergence behavior. The computation time of SILU is also good, but for finer grids it becomes more expensive than MSIMPLER.

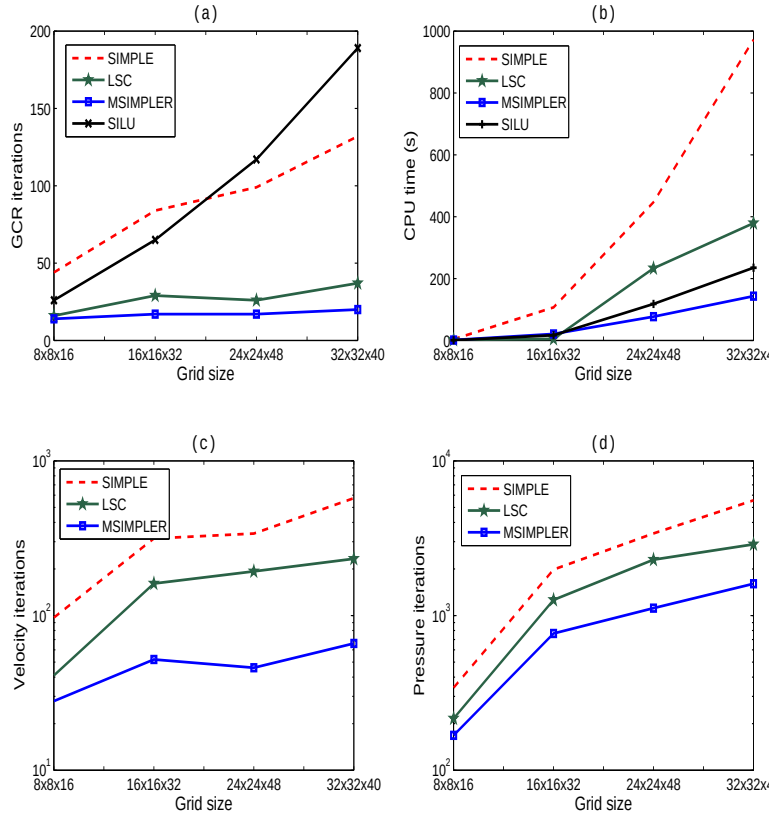


Figure 7.2: 3D Backward facing step (hexahedra): The Stokes problem is solved with accuracy 10^{-6}. PCG is used as inner solver in the block preconditioners (SEPRAN)

In the Navier-Stokes problem it is sufficient to use an accuracy of 10^{-2} per Picard step. In this case SILU performs slightly better than MSIMPLER. See Table 7.4.

To investigate the behavior of the preconditioners for tetrahedral elements we solved the 3D lid driven cavity problem (Table 7.5). For the Stokes problem the result is comparable to the hexahedral case. MSIMPLER requires less CPU time than LSC and SILU. The number of GCR iterations is almost mesh-independent.

The situation for Navier-Stokes with tetrahedra is different from that of hexahedra. Table 7.6 gives the CPU time and number of iterations. Now MSIMPLER proves to be the best choice. The increase of iterations for increasing Reynolds number is caused by an increase of the number of Picard iterations. The Reynolds dependency of all methods per Picard iteration is only mild.

Table 7.4: 3D Backward facing step (hexahedra):The Navier-Stokes problem is solved with accuracy 10^{-4}, a linear system at each Picard step is solved with accuracy 10^{-2} using preconditioned Krylov subspace methods. Bi-CGSTAB is used as inner solver in block preconditioners(SEPRAN)

Re	P_S	P_{LSC}	P_{MSR}	P_{SILU}
	GCR iter. (t_s)			Bi-CGSTAB iter. (t_s)
$8 \times 8 \times 16$				
100	200(23)	117(17.6)	74(9.6)	140(8.9)
200	314(31)	176(25)	112(14.8)	255(13.8)
400	509(47)	280(36)	168(21)	1688(49)
$16 \times 16 \times 32$				
100	447(591)	173(462)	96(162)	321(114)
200	718(839)	256(565)	145(223)	461(173)
400	1277(1223)	399(745)	235(312)	768(267)
$32 \times 32 \times 40$				
100	909(12000)	240(5490)	130(1637)	1039(1516)
200	> 1000	421(7784)	193(2251)	1378(2000)
400	> 2000	675(11000)	295(2800)	1680(2450)

Table 7.5: 3D Lid driven cavity problem (tetrahedra): The Stokes problem is solved with accuracy 10^{-6}. PCG is used as inner solver in block preconditioners (SEPRAN)

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

7.2.3 Grid Stretching

One of the unsolved issues in the iterative solution of the Navier-Stokes equations is the case of stretched grids. In practical applications it is very common to have a stretched grid, so it is important that an iterative solver is capable of dealing with such meshes. Therefore we consider a 2D lid driven cavity with a grid refined in the region where we have strong gradients. The subdivision is symmetric with respect to midpoints of the square, see Figure 7.3. The stretch factor (SF) is defined as the ratio of the largest and smallest edge in the grid.

Results for solving the Stokes problem are shown in Table 7.7. The convergence of MSIMPLER, LSC and SILU deteriorates with an increase in SF. All three preconditioners show an increase in the number of iterations with increase in stretching. For

Table 7.6: 3D Lid driven cavity problem (tetrahedra):The Navier-Stokes problem is solved with accuracy 10^{-4}, a linear system at each Picard step is solved with accuracy 10^{-2} using preconditioned Krylov subspace methods. Bi-CGSTAB is used as inner solver in block preconditioners(SEPRAN)

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

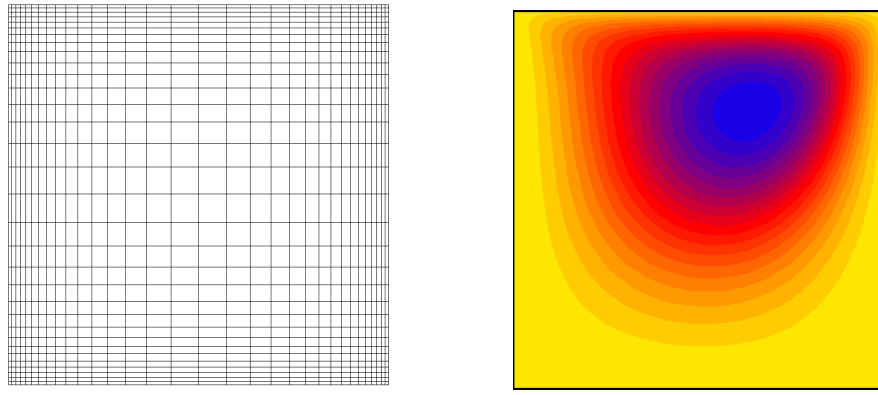


Figure 7.3: A 32×32 grid with stretch factor = 8 (Left), Streamlines plot on the stretched grid (Right)-(SEPRAN)

a large SF, the preconditioners even fail to converge. Compared to the performance of these preconditioners in the Stokes problem, the situation is even worse in the Navier-Stokes problem. Until now, as far as we know no results for LSC in stretched grids are published. In Table 7.8, we see that MSIMPLER and LSC perform poorly in stretched grids for the Navier-Stokes problem. With the increase in SF, all preconditioners mentioned show bad convergence. The inner and outer iterations in the block preconditioners stagnate after some reduction in the residual. For a certain SF, the performance of these preconditioners becomes worse with increase in Reynolds number and grid size.

Table 7.7: 2D Lid driven cavity problem on 64×64 stretched grid: The Stokes problem is solved with accuracy 10^{-6}. PCG is used as inner solver in block preconditioners (SEPRAN) .

SF	P_{LSC}	P_{MSR}	P_{SILU}
	GCR iter.	GCR iter.	Bi-CGSTAB iter.
1	20	17	96
8	49	28	189
16	71	34	317
32	97	45	414
64	145	56	NC
128	NC	81	NC

Table 7.8: 2D Lid driven cavity problem on stretched grid: The Navier-Stokes problem is solved with accuracy 10^{-4}. A linear system in each Picard step is solved with accuracy 10^{-2} using preconditioned Krylov subspace methods. Bi-CGSTAB is used as inner solver in the block preconditioners(SEPRAN).

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

7.3 IDR(s) and Bi-CGSTAB(ℓ) comparison

In this section, we compare Bi-CGSTAB (ℓ) [72] and IDR(s) preconditioned with ILU. For $\ell = 1$, we will refer Bi-CGSTAB (ℓ) as Bi-CGSTAB. In Figure 7.4, the number of iterations and CPU-time for the solution of the Stokes backward facing step problem for two different grid are plotted. We see that an increase of s from 1 to 2 reduces the CPU-time considerably (especially for the fine grid), but further increase of s has no

significant profit. The increase of s does not give a monotone decrease of iterations.

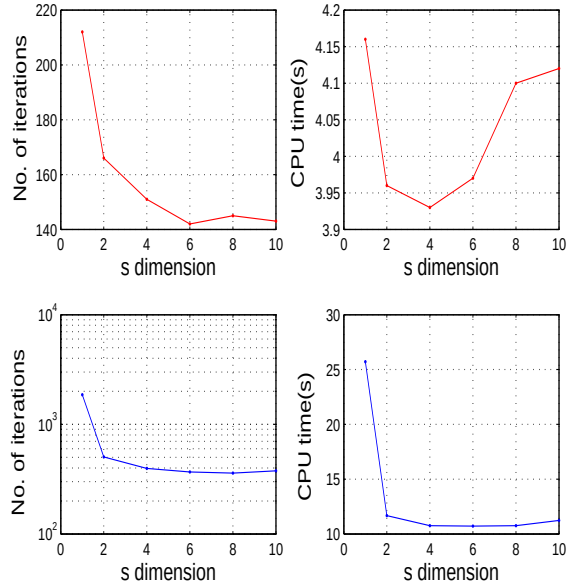


Figure 7.4: The 2D Stokes backward facing step problem solved with ILU preconditioned IDR(s) method with varying s dimension: 32×96 grid (Top), 64×96 grid (Bottom).

In Table 7.9 we see the number of matrix-vector multiplications and CPU-time for the Stokes driven cavity problem solved on a uniform grid. Since 1 iteration of Bi-CGSTAB costs 2 matrix-vector (Mat.-Vec.) multiplications and IDR(s) requires s multiplications per step this is the best way of comparison. The difference between Bi-CGSTAB and IDR(4) is not very significant. However, the table suggests that this difference increases for increasing grid size.

The main reason to use IDR(s) is it better performance in case the ILU preconditioner is not so efficient. Figure 7.5 shows the effect of the SF for the Stokes problem on a 128×128 grid. The optimal choice, IDR(7), shows a much better performance for increasing SF than Bi-CGSTAB.

IDR(s) is also compared with Bi-CGSTAB(ℓ) for more difficult problems. In Table 7.10, we see with increase in grid size and stretch factor, IDR(s) performance becomes better than Bi-CGSTAB(ℓ) for different values of s and ℓ.

In general, IDR(s) proved to be more stable than Bi-CGSTAB(ℓ). For the problems, where Bi-CGSTAB(ℓ) shows convergence, IDR(s) always shows convergence. However, in some cases, we observed divergence of Bi-CGSTAB(ℓ) -especially with $\ell = 1$ - whereas IDR(s) converged.

Table 7.9: ILU preconditioned Krylov subspace methods comparison with increasing grid size for the driven cavity Stokes flow problem.

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

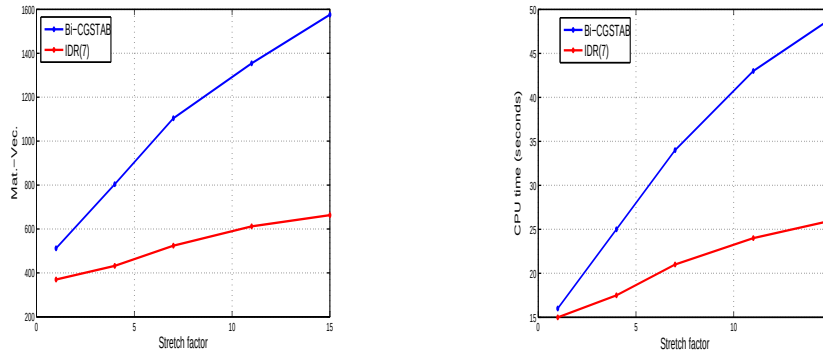


Figure 7.5: SILU preconditioned Krylov subspace methods comparison with increasing stretch factor for the driven cavity Stokes flow problem.

Table 7.10: SILU preconditioned Krylov subspace methods comparison with increasing grid size and stretch factor for the driven cavity Stokes flow problem.

Stretch factor	Bi-CGSTAB(ℓ)		IDR(s)	
	ℓ	Mat.-Vec. (t_s)	s	Mat.-Vec. (t_s)
128×128				
4	4	608(23)	6	437(19)
10	4	944(34)	6	696(28)
256×256				
4	4	1864(283)	6	1383(243)
	6	1848(285)	8	1066(204)
10	4	4184(606)	6	1927(331)
	6	3768(558)	8	1796(327)

7.4 Summary

In this chapter we studied the convergence behavior of some block preconditioners for the Stokes and Navier-Stokes problems both in 2D and 3D. Results for various grid sizes and Reynolds numbers have been investigated. We also compared the convergence with an algebraic preconditioner (SILU). Some common properties of MSIMPLER and LSC are discussed.

In all our experiments MSIMPLER proved to be cheaper than LSC. This concerns both the number of outer iterations, inner iterations and CPU time. The number of outer iterations in MSIMPLER hardly increases if a direct solver for the subsystems is replaced by an iterative solver. This is in contrast with LSC where large differences are observed. It appears that the combination of LSC with MG is almost optimal. The combination of LSC with a PCG inner solver can take many iterations and much CPU time. When problems are solved with low accuracy, for example in case of Navier-Stokes, SILU sometimes shows better performance than the other preconditioners. MSIMPLER proved to be cheaper than SILU, especially when the problem is solved with high accuracy. The performance of all these preconditioners is affected by grid stretching. The number of iterations increases with an increase in stretching or even diverges in some cases.

A newly developed Krylov method, IDR(s), preconditioned with SILU is compared with Bi-CGSTAB(ℓ). For equidistance grids, perfomance of both Krylov methods is comparable. However, when the grid is stretched and preconditioner becomes less efficient, IDR(s) start to perform better than Bi-CGSTAB(ℓ) for optimal values of s and ℓ.

Chapter 8

On iterative methods for the incompressible Stokes problem with varying viscosity

In this chapter, we discuss various techniques for solving the system of linear equations that arise from the discretization of the incompressible Stokes equations by the finite element method. The proposed solution methods, based on a suitable approximation of the Schur complement matrix, are shown to be very effective for a variety of problems. We discuss two iterative methods. These approaches use the pressure mass matrix as preconditioner (or an approximation) to the Schur complement. We compare these methods with the preconditioners (MSIMPLER, LSC) that are already discussed.

8.1 Iterative methods for the Stokes problem

In this chapter, we concentrate on the Stokes problem only. In contrast to many of the Navier-Stokes preconditioners, we do not assume that the viscosity, ν, is constant, nor that it is smaller than one. Our main goal is to investigate block preconditioners that are suitable for the Stokes problem, both for constant and varying viscosity.

In this section, we discuss techniques for efficient solution of the Stokes problem. These techniques use Krylov subspace methods on both the system and subsystem level with appropriate choices of preconditioners and approximations of the pressure Schur complement matrix. The schemes are:

1. Block triangular preconditioner with GCR

2. The Schur method (a new approach)

The first approach consists of a preconditioner that accelerates GCR to solve the Stokes problem. The second technique is an iterative method that uses Krylov methods on subsystem level. The important feature of the second method is the preconditioner used to solve the implicitly constructed Schur complement matrix $BF^{-1}B^T$ (available in matrix-vector product form) at each step of the pressure subsystem solve.

8.1.1 Block triangular preconditioner

We have already discussed both the general form (4.6) as well as the eigenvalue analysis of the block triangular preconditioner in Section 4.3. In Chapters 4 and 7, we have used a block triangular preconditioner (LSC) for solving the incompressible (Navier-) Stokes problem with constant viscosity. In this section, a special block triangular preconditioner is discussed for the Stokes problem with an approximation of the Schur complement matrix that enhances the convergence of the Krylov methods.

For the Stokes problem, with constant viscosity, the pressure mass matrix Q_p is known to be a cheap and spectrally equivalent approximation to the Schur complement matrix [70], [35]. In [70] $\tilde{S} = -Q_p$ is used with block diagonal preconditioner ($\mathcal{D}_b$) for the Stokes problem. The block diagonal preconditioner is symmetric and positive definite and can be used as a preconditioner for MINRES.

In our work, we use the pressure mass matrix with a block triangular preconditioner given by:

$$P_t = \begin{bmatrix} F & B^T \\ 0 & -Q_p \end{bmatrix}. \tag{8.1}$$

Algorithm 4.2 describes the steps that are used to solve $P_t z = r$. In general it is not a good idea to use a nonsymmetric preconditioner for a symmetric problem. However, if we use a block triangular preconditioner the number of iterations is half the number of iterations necessary when a block diagonal preconditioner is used [34], whereas the increase in cost per iteration is only minimal.

If the Stokes problem is scaled with a constant viscosity ν, then the pressure mass matrix is also scaled with a constant viscosity $(1/\nu)Q_p$. The effectiveness of this approach is justified by the following theorems from [35, p. 270].

Theorem 8.1.1. *For any flow problem with Dirichlet boundary conditions (prescribed velocities) discretized using a uniformly stable mixed approximation on a shape regular, quasi-uniform subdivision of* $\mathbb{R}^2$*, the pressure Schur complement matrix* $BF^{-1}B^T$ *is spectrally equivalent to the pressure mass matrix* Q_p*, with*

$$\mu^2 \leq \frac{\langle BF^{-1}B^T\mathbf{q},\ \mathbf{q}\rangle}{\langle Q_p\mathbf{q},\ \mathbf{q}\rangle} \leq 1 \ \textit{for all}\ \mathbf{q} \in \mathbb{R}^m,\ \mathbf{q} \neq 0. \tag{8.2}$$

The inf-sup constant μ *is bounded away from zero independently of h, and the condition number satisfies* $k(BF^{-1}B^T) \leq C/(c\mu^2)$*, where C and c are the constants defined by*

$$ch^2 \leq \frac{\langle Q_p\mathbf{q},\ \mathbf{q}\rangle}{\langle \mathbf{q},\ \mathbf{q}\rangle} \leq Ch^2 \ \textit{for all}\ \mathbf{q} \in \mathbb{R}^m,\ \ \mathbf{q} \neq 0. \tag{8.3}$$

Similar bounds also exist for the Neumann boundary condition problem. An extra condition $\mathbf{q} \neq 1$ is required in (8.2) and (8.3) in case of enclosed flow, because the vector $\begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix}$ corresponds to an eigenvector for a zero eigenvalue of the Stokes matrix.

Theorem 8.1.2. *If the Stokes problem is preconditioned with a block triangular preconditioner, then the preconditioned system has eigenvalue $\lambda = 1$ of multiplicity n_u and the remaining eigenvalues depend on the approximation to the Schur complement matrix.*

In this paper, we accelerate GCR with a block triangular preconditioner that uses the pressure mass matrix approximation for the Schur complement matrix. For simplicity, we call this approach PMM. The spectral equivalence of the Schur complement matrix for the variable viscosity Stokes problem and the variable viscosity pressure mass matrix has been proved recently by Olshanskii [57].

8.1.2 The Schur method

The Schur method is based on the block factorization of problem (2.21). The Schur complement matrix, $(BF^{-1}B^T)$, present in the factorization is treated implicitly. In order to apply pressure-correction type methods, we split the coefficient matrix as follows:

$$\begin{bmatrix} F & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix} = \begin{bmatrix} F & 0 \\ B & -BF^{-1}B^T \end{bmatrix} \begin{bmatrix} u^* \\ \delta p \end{bmatrix}, \tag{8.4}$$

where

$$\begin{bmatrix} u^* \\ \delta p \end{bmatrix} = \begin{bmatrix} I & F^{-1}B^T \\ 0 & I \end{bmatrix} \begin{bmatrix} u \\ p \end{bmatrix}. \tag{8.5}$$

Then the systems of equations can be solved in the steps given in Algorithm 8.1.

As already mentioned, the Stokes problem is symmetric and indefinite. However, the subsystems corresponding to the velocity (vector Poisson) and pressure in Steps 1, 2 and 3 are symmetric and definite. F^{-1} in Steps 1 to 3 is computed approximately by solving the velocity subsystem with an inexact solver. The best option is to use an MG preconditioned CG or some multigrid technique since both of these methods are known to give optimal convergence for Poisson-type problems.

The pressure subsystem in Step 2 can in principle be solved efficiently by CG. However, we do not construct $BF^{-1}B^T$ explicitly, but approximately solve $(-BF^{-1}B^T)p_\delta = r_p - Bu_f$ within each step of CG. Since F^{-1} is computed inexactly, CG can only be applied if the number of iterations used to do this is kept constant in each step. This is due to the fact that CG requires a constant matrix as preconditioner. This problem can be overcome by either using a stand-alone solver, such as multigrid, or flexible Krylov methods (GCR in our case). The efficiency of the Schur method requires efficient treatment of Step 2 . Because the Schur complement matrix is not constructed explicitly, we need a special type of preconditioner. The pressure mass matrix appears to be an efficient preconditioner for the Schur subsystem. If we use the same accuracy to solve the system with PMM and to solve Step 2 of the Schur method, the number of pressure mass matrix preconditioned GCR iterations in both methods is almost the

Algorithm 8.1 The Schur method

Initialize $u^{(0)}$, $p^{(0)}$ and *maxiter* (maximum iterations)
Compute: $r_u = f - Fu^{(0)} - B^T p^{(0)}$
$r_p = g - Bu^{(0)}$
For $k = 0$ to *maxiter*

1. Solve $Fu_f = r_u$
2. Solve $-BF^{-1}B^T p_\delta = r_p - Bu_f$
3. Update $u_\delta = u_f - u_l$, where u_l is obtained by solving $Fu_l = B^T p_\delta$
4. Update $u^{(k+1)} = u^{(k)} + u_\delta$
5. Update $p^{(k+1)} = p^{(k)} + p_\delta$
6. Update $r_u = f - Fu^{(k+1)} - B^T p^{(k+1)}$
7. Update $r_p = g - Bu^{(k+1)}$
8. **If** converged **Exit**

End For

same. These iterations govern the efficiency of both techniques. This observation also motivates the use of GCR instead of flexible CG [56].

Based on the Schur method (Algorithm 8.1), we propose two schemes:

1. **Schur method as direct method:** By requiring the accuracy of the subsystem solves to be the same or higher than the outer accuracy, the Schur method can be used as a direct solver. To solve the Schur pressure system $(-BF^{-1}B^T)p_\delta = r_p - Bu_f$, we use the pressure mass matrix Q_p as preconditioner. The system $(-BF^{-1}B^T)p_\delta = r_p - Bu_f$ is solved with the help of GCR in which preconditioned matrix-vector products within $(BF^{-1}B^T)p_\delta$ are obtained by computing preconditioned residual $S^{(k+1)} = Bu_m$, where u_m is obtained by solving a subsystem $Fu_m = B^T {Q_p}^{-1} r^{(k)}$ and $r^{(k)}$ is the residual computed in the previous GCR iteration.

2. **Schur as an iterative method:** When the inner systems are solved with a lower accuracy than desired for the final solution, outer iterations are required. Again, the pressure mass matrix is used as preconditioner for the pressure subsystem.

From the above discussion, it is clear that three subsystems are solved for the velocity unknowns u_f, u_m, and u_l and one subsystem is solved for the pressure unknown p_δ in each iteration of the Schur method. The most expensive part of the algorithm is the computation of u_m since the number of times $Fu_m = r$ must be solved is equal to the total number of GCR iterations that are required to solve the pressure subsystem.

One of the advantages that can be seen from the above algorithm is that most of the computations are done at subsystem level. The system level computations can be reduced by a proper choice of inner accuracy. For example, if subsystems are solved as accurately as the outer tolerance requires, only one outer iteration is required. However, more outer iterations are typically required. This will be further discussed in Section 8.3, based on some numerical experiments.

8.1.3 Variant of LSC

In the LSC method the velocity mass matrix is used in the approximation of the Schur complement matrix. In a recent paper [50], a scaling matrix has been constructed that improves the convergence considerably in the case of large viscosity contrasts. The scaling is based on the maximum row entry in the velocity matrix, $(M_1)_{ii} = (M_2)_{ii} = max_j|F_{ij}|$. Based on this scaling, the LSC preconditioner is used to solve the Stokes problem with sharp viscosity contrasts discretized with Q1-P0 elements. Because these elements do not satisfy a discrete LBB condition [35], it may be necessary to modify the discretized equations by adding a stabilization term to the continuity equations. If such a modification is made, an appropriate adaptation of the approximate Schur complement is also needed. In this chapter, we also use essentially the same scaling, by using the diagonal of the velocity matrix ($M_1 = M_2 = diag(F)$). For simplicity, we call this approach LSC_D.

8.1.4 Construction of variable viscosity pressure mass matrix

The standard pressure mass matrix is defined independently of the viscosity

$$(Q_p)_{i,j} = \int_\Omega \phi_i \phi_j d\Omega, \tag{8.6}$$

where ϕ_j and ϕ_i are the standard finite element basis functions for the pressure.

Experiments show that solving the variable viscosity problem using this matrix as a preconditioner gives slow convergence. The Schur complement matrix, however, contains F^{-1}, which means that it is proportional to the inverse of the viscosity. So, it makes sense to scale the pressure mass matrix used as a preconditioner by the inverse of the viscosity. In this section, we discuss the construction of the pressure mass matrix and scaling with the viscosity.
In case of constant viscosity, scaling of the pressure mass matrix is trivial. However, in case of variable viscosity, such scaling must be done carefully. Here, we assume that the viscosity is available at each point of the grid, and consider two alternatives to incorporate the scaling.

1. Explicit scaling of the pressure mass matrix, which implies pre and post multiplication of Q_p by a diagonal matrix S_ν: $Q_{pe} = S_\nu^{-1} Q_p S_\nu^{-1}$, where $S_\nu = diag(\sqrt{\nu})$. This guarantees that the mass matrix remains symmetric. The evaluation of ν is done either on grid points (Taylor-Hood elements) or on elements (Crouzeix-Raviart).

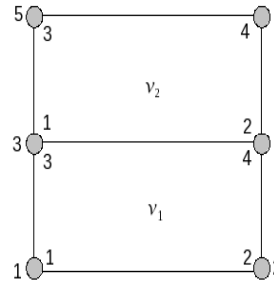


Figure 8.1: A grid with 2 elements.

2. Implicitly scaling the pressure mass matrix can be done at the time of formation of the pressure mass matrix,

$$(Q_{pi})_{i,j} = \int_{\Omega} (1/\nu)\phi_i\phi_j d\Omega. \tag{8.7}$$

If the viscosity is defined to be piecewise constant on each element, we have multiple viscosity values to choose from at each node on the interface of two or more regions if we choose to scale Q_p explicitly. Consider the grid consisting of two elements shown in Figure 8.1 with viscosity ν_1 in element 1 and ν_2 in element 2. We can see that nodes 3 and 4 of element 1 and nodes 1 and 2 of element 2 (using local numbering) are common to both elements. So, there are two different values of the viscosity that could be used at each node. Before constructing the global mass matrix, we define

$$\alpha_{ij}^k = \int_{e_k} \phi_i\phi_j dA_k, \tag{8.8}$$

where α_{ij}^k represents the contribution to $(Q_{pi})_{i,j}$ from element k where i and j represent node indices. The global pressure mass matrix from these two elements using implicit scaling is:

$$Q_{pi} = \begin{bmatrix} \nu_1^{-1}\alpha_{11}^1 & \nu_1^{-1}\alpha_{12}^1 & \nu_1^{-1}\alpha_{13}^1 & \nu_1^{-1}\alpha_{14}^1 & 0 & 0 \\ \nu_1^{-1}\alpha_{21}^1 & \nu_1^{-1}\alpha_{22}^1 & \nu_1^{-1}\alpha_{23}^1 & \nu_1^{-1}\alpha_{24}^1 & 0 & 0 \\ \nu_1^{-1}\alpha_{31}^1 & \nu_1^{-1}\alpha_{32}^1 & \nu_1^{-1}\alpha_{33}^1 + \nu_2^{-1}\alpha_{11}^2 & \nu_1^{-1}\alpha_{34}^1 + \nu_2^{-1}\alpha_{12}^2 & \nu_2^{-1}\alpha_{13}^2 & \nu_2^{-1}\alpha_{14}^2 \\ \nu_1^{-1}\alpha_{41}^1 & \nu_1^{-1}\alpha_{42}^1 & \nu_1^{-1}\alpha_{43}^1 + \nu_2^{-1}\alpha_{21}^2 & \nu_1^{-1}\alpha_{44}^1 + \nu_2^{-1}\alpha_{22}^2 & \nu_2^{-1}\alpha_{23}^2 & \nu_2^{-1}\alpha_{24}^2 \\ 0 & 0 & \nu_2^{-1}\alpha_{31}^2 & \nu_2^{-1}\alpha_{32}^2 & \nu_2^{-1}\alpha_{33}^2 & \nu_2^{-1}\alpha_{34}^2 \\ 0 & 0 & \nu_2^{-1}\alpha_{41}^2 & \nu_2^{-1}\alpha_{42}^2 & \nu_2^{-1}\alpha_{43}^2 & \nu_2^{-1}\alpha_{44}^2 \end{bmatrix} \tag{8.9}$$

In (8.9), it is evident that for a high viscosity contrast, the smaller value of ν will dominate the definition of Q_{pi} (due to its inversion) at the nodes that are shared by these two elements, and large entries will be observed on the diagonal of the matrix.

8.2 Convergence issues

We consider two variable viscosity problems:

- **Extrusion Problem with a variable viscosity:** The Stokes problem is solved in the two dimensional domain shown on the left of Figure 8.2. The problem we consider is that of a round aluminum rod, which is heated and pressed through a die. In this way, a prescribed shape can be constructed. In this specific example, we consider the simple case of the construction of a small round rod. The viscosity model used describes the viscosity as function of shear stress and temperature, which are highest at the die where the aluminum is forced to flow into a much smaller region. The end rod is cut, which is modeled as a free surface. Boundary conditions are given by prescribed velocity at the inlet and a stress free condition at the outlet. At the container surface (boundary of thick rod), we have a no slip condition. At the die, we have friction, which is modeled as a slip condition along the tangential direction and a no flow condition in the normal direction. The round boundary of the small rod satisfies free slip in the tangential direction and no flow in the normal direction.

- **Geodynamic problem having a sharp viscosity contrast:** This problem is a benchmark problem known as the SINKER model in [50]. It models a geodynamic flow on a square region. Inside the region is a square with a different (but constant) viscosity and jump in density, resulting in a sharp viscosity contrast. The configuration is shown in [50] and in Figure 8.2. Boundary conditions are no normal flow and no shear stress (at all boundaries). For this boundary condition, pressure can be determined only up to an arbitrary additive constant.

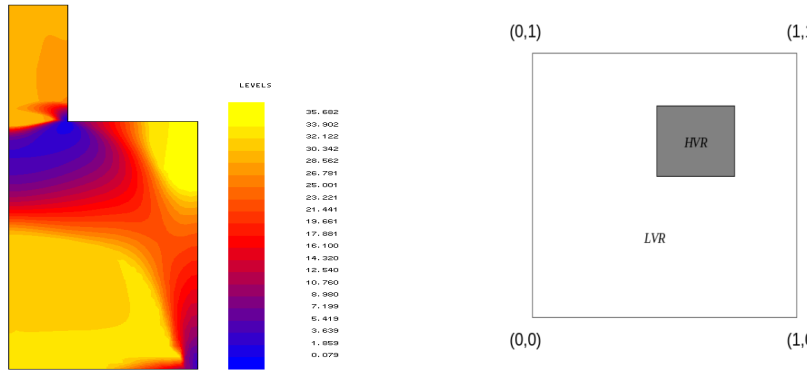


Figure 8.2: Two dimensional domain for the variable viscosity Stokes problem (Left). At right, a 2D geodynamics test model: LVR represents the low viscosity region with density $\rho_1 = 1$ and viscosity $\nu_1 = 1$, and HVR denotes the high viscosity region with density $\rho_2 = 2$, and constant viscosity ν_2 (1, 10^3 and 10^6).

If we try to solve the SINKER problem with high viscosity contrast (e.g 10^6) and use an inexact solver, e.g ICCG(0), for the subsystem solves, we fail to get convergence due to stagnation of the inner solvers for high accuracies (10^{-6}). Each time, a suitable tolerance is determined for which the inner solver shows convergence. We would like to use higher accuracies to solve the subsystems because, for high viscosity contrast problems, the number of iterations of the outer Krylov method depends on the inner accuracies. Another issue with high viscosity contrast problems is that if we use convergence criteria based on the L_2 norm, some preconditioners e.g. PMM, lead to fewer iterations. However, an inaccurate solution is obtained with this convergence criterion. In order to achieve suitably accurate results, we must require a much higher accuracy from the Krylov method, which is not a good practice. For example if we use a direct solver for the high viscosity contrast problem PMM requires much less iterations than LSC. However, the results with PMM are less accurate than those computed by LSC_D (see Section 8.1.3) for the same tolerance. Figure 8.3 shows the pressure in the high viscosity region of the SINKER model for PMM and LSC_D. The differences in these solutions can be easily seen. Note also that at other places the difference is not visible. It has been observed that the solution obtained with LSC_D mimics that obtained with a direct solver.

Remark 8.2.1. *If we use a preconditioner for the Schur complement that involves the diagonal of the velocity matrix D^{-1}, the error in the iterative method using a direct method for the subsystems becomes small. This has been verified for LSC_D, $BD^{-1}B^T$ and SIMPLE.*

The above remark is true even if convergence for the velocity and pressure subsystems is achieved with an iterative method, without scaling.

To overcome the issue with convergence of the subsystems and accuracy of the solution, we scale the original problem. In our case, we use a variant of scaling that has already been used with SIMPLE preconditioners [88]. S_m is a scaling matrix given by:

$$S_m = \begin{bmatrix} \sqrt{diag(F)} & 0 \\ 0 & \sqrt{diag(BD^{-1}B^T)} \end{bmatrix}. \tag{8.10}$$

This scaling is applied to the complete system ($\mathcal{A}x = b$) and to the velocity and pressure subsystems (within the preconditioner), but is only well-defined when $F_{ii} > 0$ and $(BD^{-1}B^T)_{ii} > 0$. In our case, both conditions are satisfied and, hence, the matrix S_m is well-defined. With this scaling, convergence criteria are based on the residual in a scaled L_2 norm.

In our case, the linear system (for the SINKER model) is scaled before the iterative methods are employed. After scaling, the pressure obtained with PMM mimics the solution obtained with the exact solver. With this scaling, the number of iterations required for convergence by the preconditioned iterative method may increase. Scaling, however, only slightly changes the eigenvalue spectrum of the preconditioned system. This has been proven for the diffusion problem having extreme contrasts (up to 10^7) in the coefficients using ICCG [90]. For example, if we consider a preconditioned system, $P^{-1}\mathcal{A}x = P^{-1}b$, and if both the preconditioner and the coefficient matrix are scaled with the same matrix, D_s, the scaled matrices become $\hat{P} = D_s^{-1}P$ and $\hat{\mathcal{A}} = D_s^{-1}\mathcal{A}$, and the preconditioned system matrix after scaling is given by: $\hat{P}^{-1}\hat{\mathcal{A}} = P^{-1}D_sD_s^{-1}\mathcal{A} = P^{-1}\mathcal{A}$. So the spectrum of the preconditioned scaled system is the same as the preconditioned unscaled system.

Figure 8.4 shows the computed eigenvalue spectrum before and after scaling. The spectrum of the preconditioned system clearly remains almost unchanged. If we use scaling, so that our system, $\mathcal{A}x = b$ becomes $S_m^{-1}\mathcal{A}S_m^{-1}S_mx = S_m^{-1}b$, the most important change is the termination criterion for the iterative method. The relative convergence criteria for both the scaled and unscaled problems are as follows:

For the unscaled system

$$||b - \mathcal{A}\hat{x}||_2 \leq ||b||_2\, tol, \tag{8.11}$$

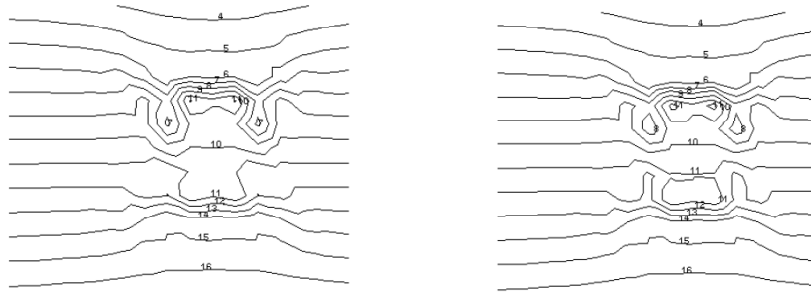
where $\hat{x}$ is an approximate solution and *tol* is the desired tolerance.

For the scaled system, the convergence criterion will be:

$$||S_m^{-1}b - S_m^{-1}\mathcal{A}S_m^{-1}\hat{y}||_2 \leq ||S_m^{-1}b||_2\, tol, \tag{8.12}$$

where $\hat{x} = S_m^{-1}\hat{y}$.

From the discussion above, it is clear that to get convergence (in high viscosity contrast problems), scaling subsystems in the preconditioner is required. However, this does not guarantee that the computed solution will be accurate. For an accurate solution, the complete system also requires scaling with an appropriate scaling operator (S_m in our case).



(a) PMM (b) LSC_D

Figure 8.3: Solution of the variable viscosity Stokes problem using various solution schemes: The plot shows the pressure solution in the high viscosity region at the SINKER problem.

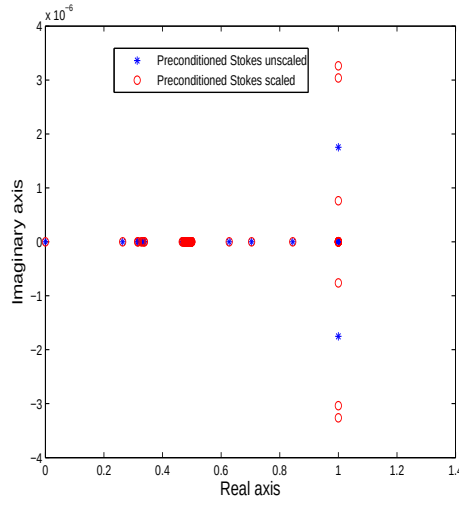


Figure 8.4: Eigenvalue spectrum of the Stokes problem.

The error between the results obtained with PMM and other preconditioners is more visible in the SINKER model. However, we also want to check whether this error also exists in case of more general problems. It has been observed that for the same preconditioner, different implementations (Split, left or right preconditioner) leads to the same eigenvalue spectrum of the preconditioned system [83, p. 175]. Therefore, the same number of iterations are expected for a desired accuracy. In our software we use GCR in combination with a right preconditioner. This suggests that the stopping criterion is independent of the preconditioner. The convergence criterion we use is given in (8.11). From the relation in (8.11) it can be proved that

$$\frac{\|x - x_k\|_2}{\|x\|_2} \leq K_2(\mathcal{A}) \frac{\|r_k\|_2}{\|b\|_2} \leq K_2(\mathcal{A})\ tol.$$

This implies that the relative error depends on the condition number of the matrix. The right preconditioner has the advantage that it only effects the operator and not the right-hand side. This implies that stopping criteria for various preconditioners are the same.

Remark 8.2.2. *Since in right preconditioning we solve $AP^{-1}y = b$ with $x = P^{-1}y$. One must be careful with stopping criteria that are based on the error $\|y - y_k\|_2$ because this error may be much smaller than the error norm $\|x - x_k\|_2$ (equal to $\left\|P^{-1}(y - y_k)\right\|_2$) [83, p. 175].*

However, if we solve the Stokes problem in combination with the PMM preconditioner, we can see that for increasing grid size the actual error becomes much larger than the required accuracy (see Table 8.1 and 8.2). This has already been observed in the SINKER problem. Therefore we study the stopping criterion in more detail.

In case of left preconditioning, the convergence criterion depends also on the preconditioner

$$\left\|P^{-1}(b - \mathcal{A}x_k)\right\|_2 < \left\|P^{-1}b\right\|_2\ tol.$$

Now the condition number is changed to $K_2(P^{-1}\mathcal{A})$ which is expected to be smaller than $K_2(\mathcal{A})$ and the convergence criterion may be quite different then the criterion based on the unpreconditioned residual.

To get reasonable accuracy in case of right preconditioner, scaling a complete system may be a better option to use. The idea is to minimize the condition number of the matrix to $K_2(S_m^{-1}\mathcal{A})$, where S_m is the scaling matrix (already been tested in SINKER). Scaling the complete system results in a termination based on $\left\|S_m^{-1}r_k\right\|$. This gives an increase in iterations, but clearly much more reliable result. Diagonal scaling is considered to be optimal if the original system is SPD [80]. For the symmetric indefinite system we also use a diagonal scaling matrix. Because scaling of the system only changes the convergence criterion, it is much cheaper to scale only the residual by S_m^{-1}, before checking the accuracy. Some cheap options are:

$$S_{m1} = \begin{bmatrix} diag(F) & 0 \\ 0 & diag(BD^{-1}B^T) \end{bmatrix}. \tag{8.13}$$

Another choice of scaling the stopping criteria can be that $diag(BD^{-1}B^T)$ is replaced by $diag(Q_p)$ given as:

$$S_{m2} = \begin{bmatrix} diag(F) & 0 \\ 0 & diag(Q_p) \end{bmatrix}. \tag{8.14}$$

For the pressure part, the pressure mass matrix is a good scaling operator because in case of a lumped pressure mass matrix, this is only a diagonal matrix.

Tables 8.1 to 8.3 compare the error in the Stokes problem preconditioned with PMM and LSC with different termination criteria. It is clear that, if we base the termination on the norm of the residual, the real error increases for increasing grid size. On the other hand if we use the norm of the residual multiplied by the inverse of the preconditioner, we see that the real error is almost constant. Of course the number of iterations increases in this case. $P^{-1}r_k$ is available from the previous step and in case of variable preconditioner, the operation $P^{-1}r_k$ at each step is no more constant. This may rise to a less accurate residual than constant scaling, if subsystems are solved with small accuracy. The real error in all these cases is computed by solving the original system with a direct solver.

Table 8.4 shows the results of this approach for two different choices of the scaling matrices, S_{m1} and S_{m2} . It is clear that this cheap approach is a very good alternative for complete scaling. Both scaling matrices are comparable, S_{m1} gives slightly better error estimate than S_{m2}. For more information concerning stopping criteria we refer to [91], [2].

Table 8.1: Backward facing step Stokes problem (PMM preconditioner)

Grid	$\|r_k\|$		$\|P^{-1}r_k\|$		$\|S_m^{-1}r_k\|$	
	Iter.	Error-p	Iter.	Error-p	Iter.	Error-p
8×24	18	2e-5	22	2e-7	20	2e-6
16×48	17	3e-4	24	8e-7	20	4e-6
32×96	16	1e-3	23	5e-7	20	1e-5

Table 8.2: Driven cavity Stokes problem (PMM preconditioner)

Grid	$\|r_k\|$		$\|P^{-1}r_k\|$		$\|S_m^{-1}r_k\|$	
	Iter.	Error-p	Iter.	Error-p	Iter.	Error-p
16×16	9	9e-5	15	9e-8	13	9e-7
32×32	9	3e-4	16	7e-8	13	1e-6
64×64	9	2e-3	16	9e-8	13	2e-6
128×128	8	4e-3	16	1e-7	12	8e-6

Table 8.3: Driven cavity Stokes problem (LSC preconditioner)

Grid	$\|r_k\|$		$\|P^{-1}r_k\|$		$\|S_{m1}^{-1}r_k\|$	
	Iter.	Error-p	Iter.	Error-p	Iter.	Error-p
64×64	10	1e-4	19	1e-8	15	3e-7
128×128	11	4e-4	24	1e-8	19	8e-7

Table 8.4: Driven cavity Stokes problem solved using scaled stopping criteria.

Grid	PMM				LSC			
	$\|S_{m1}^{-1}r_k\|$		$\|S_{m2}^{-1}r_k\|$		$\|S_{m1}^{-1}r_k\|$		$\|S_{m2}^{-1}r_k\|$	
	Iter.	Error-p	Iter.	Error-p	Iter.	Error-p	Iter.	Error-p
16×16	16	1e-8	15	9e-8	11	2e-8	11	2e-8
32×32	17	2e-8	16	7e-8	14	2e-8	13	2e-7
64×64	17	4e-8	16	9e-8	19	1e-8	17	7e-8
128×128	18	2e-8	16	1e-7	24	1e-8	22	7e-8

8.2.1 Scaling of the velocity mass matrix

We use AMG/CG to solve the subsystems in the block preconditioners. One of the properties of the AMG method that we use is that it requires a constant number of unknowns per grid point. If the boundary conditions for the velocity are such that only a subset of the degrees of freedom is prescribed, we must define an approximation to the boundary that includes all velocity components. This is, for example, the case if the normal velocity component is prescribed in combination with the shear stress. The normal velocity is usually eliminated, leading to only 1(2D) or 2(3D) degrees of freedom on those points. To overcome this problem, we approximate the given normal velocity component by a mixed boundary condition of the form $c_n u_n + \sigma^{nt} = c_n \bar{u}$, where c_n a large number and $\bar{u}$ is the prescribed value of the normal velocity and σ^{nt} is the tangential component of the stress tensor.

Such an approximation works well except when the velocity mass matrix is used within the preconditioner, as in the Schur complement approximation of the MSIMPLER preconditioner. In that case, the inverse of the diagonal of the velocity mass matrix is used as a scaling matrix. The combination of this preconditioner with the approximate boundary condition leads to a lack of convergence in the outer Krylov method. To overcome this difficulty, we update the velocity mass matrix by multiplying the entries corresponding to the boundary elements with the approximate boundary conditions by a factor of c_n. After this change MSIMPLER converges as quickly as it does in the case of full Dirichlet boundary conditions.

In our experiments, we observe no significant change in the number of outer/inner iterations required using either the approximate or exact boundary conditions (see Figure 8.5). The same observation is true if LSC is used as preconditioner.

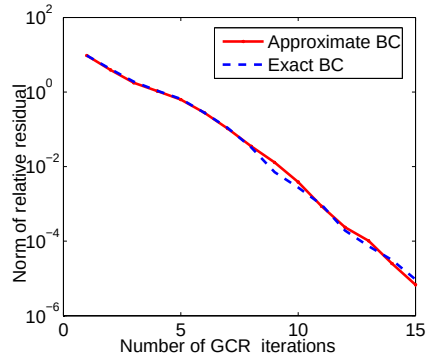


Figure 8.5: Convergence of MSIMPLER preconditioned GCR, where the subsystems are solved with ICCG(0).

8.3 Numerical experiments

Numerical experiments are performed for both constant and variable viscosity problems. The constant viscosity problem that is solved is Test Case 1. The variable viscosity problems are the extrusion and the SINKER problems discussed before.

In the tables that follow, we use the notation Schur(eps) to denote that all velocity and pressure subsystems are solved with an accuracy of 10^{-eps}. The term "Iter." gives the number of outer iterations required in the Schur method and preconditioned GCR, while "inner" refers to the total number of inner iterations needed to solve the Schur complement system in the Schur method. PMM stands for the block triangular preconditioner that uses the pressure mass matrix as preconditioner for the pressure part. The Stokes problem is solved up to an accuracy of 10^{-6}. The iteration is stopped if the current iterate satisfies the inequality $\frac{\|r^k\|_2}{\|b\|_2} \leq tol$, where r^k is the residual at the kth step of Krylov subspace method, b is the right-hand side, and tol is the desired tolerance value.

To solve subsystems, we use an exact (direct) solver, ICCG(0) and AMG (algebraic multigrid) preconditioned CG from the library ML (Multi-Level Preconditioning Package) [46]. The choice for multigrid is based on its optimal convergence for Poisson-type problems. Smooth aggregated multigrid, as is implemented in ML, is known to be particularly effective for the vector Poisson problems of interest here.

8.3.1 Isoviscous problem

In this section, we consider the solution of the driven cavity Stokes problem with a constant viscosity. For PMM, LSC and LSC_D, we use an accuracy of 10^{-2} for the velocity subsystem and 10^{-1} for the pressure subsystem and vice versa for MSIMPLER. We start with 2 level AMG for a 32×32 elements problem and increment the number of levels when the grid size is doubled in each direction. In Table 8.5, we

report the number of outer iterations and CPU time required for solution. The table shows that PMM and the Schur method both scale well as the problem size increases. Scaling of PMM has also been observed for problem with up to 10^8 degrees of freedom [39]. Moreover, the CPU time consumed by each of these two approaches is less than that needed for LSC-type preconditioners and MSIMPLER. One iteration of the Schur method, however, is more expensive than the other preconditioned Krylov methods. In Figure 8.6, the number of iterations required for the pressure and velocity subsystems is plotted. The increase in the number of AMG/CG iterations required - as the grid size (and , so, the number of levels) increases- to solve the velocity subsystem is smaller for PMM and the Schur method than it is for the other methods. Figure 8.7 shows the number of AMG/CG iterations required per outer iteration for the velocity subsystem, for a number of block preconditioners. We see a small increase as the grid size increases, but this is not significant on the finer grids.

If we use the Schur method as direct method (Schur(6)), we require only one pressure step. However, since the pressure matrix requires inversion of the velocity matrix, we use an ICCG(0) to perform $Pz = r$ step in outer GCR iteration and AMG/CG to solve the velocity subsystems with high accuracy. The iterative approach with the Schur method (Schur(1)) requires more outer iterations, but the total number of velocity iterations is smaller than for the direct method, due to the lower accuracy. So, for finer grids, Schur(1) is less time consuming than Schur(6).

From Table 8.5, we can conclude that, of the methods compared, PMM shows the best convergence behavior for the Stokes driven cavity problem. It is also evident from results that LSC_D performs only well for linear elements [50].

Table 8.5: The Stokes driven cavity flow problem with Q2-Q1 discretization(SEPRAN) with AMG/CG for the velocity subsystem solves and ICCG(0) for the Schur subsystem solves. Solution accuracy is 10^{-6}.

Preconditioner	Grids			
	32×32	64×64	128×128	256×256
	Iter. (time in seconds)			
PMM	11(1.4)	10(5.6)	9(23.6)	9(97)
LSC	10(1.38)	13(8.3)	17(54)	22(319)
LSC_D	22(3.2)	37(25)	80(275)	180(2880)
MSIMPLER	13(1.5)	16(8)	22(50)	29(300)
Schur(1)	6(3)	5(10.2)	5(46)	6(221)
Schur(6)	1(2)	1(10.6)	1(53)	1(251)

8.3.2 Extrusion problem with a variable viscosity

The next problem that we consider is the extrusion problem, which has a smooth variable viscosity. For this problem, we make a comparison between the Schur method and the various preconditioners. We use AMG/CG to solve the velocity subsystem

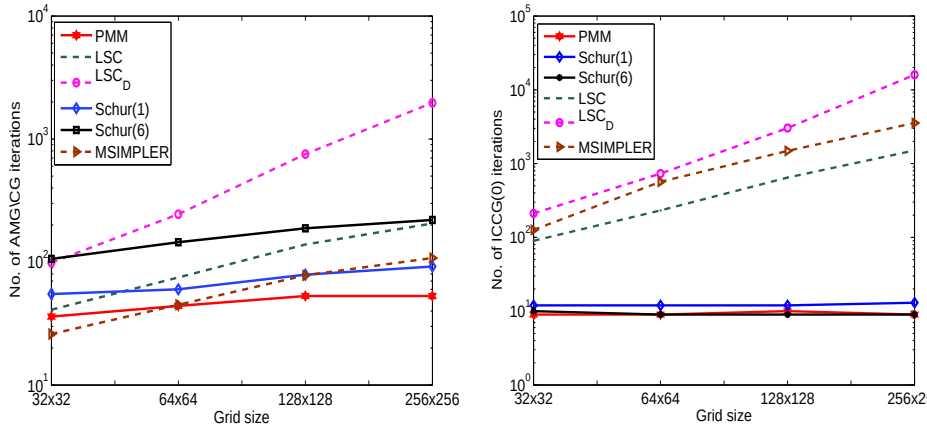


Figure 8.6: Solving constant viscosity Stokes problem with accuracy 10^{-6}: At left, the total number of iterations required for the velocity subsystem. At right, the total number of iterations required for the pressure subsystem.

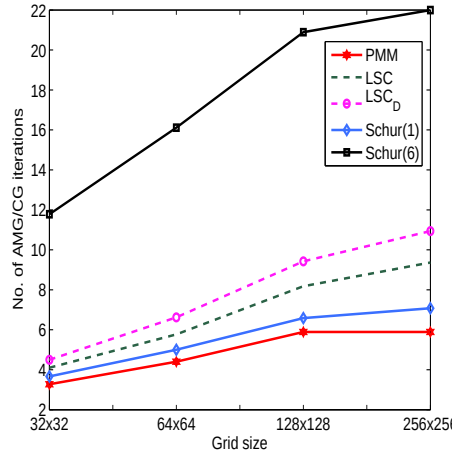


Figure 8.7: Number of AMG/CG iterations required to solve the velocity subsystem at each iteration of the iterative method.

in all iterative methods. In MSIMPLER, LSC and LSC_D only, AMG/CG is also employed for solving the pressure subsystems as for PMM or the Schur method only a few iterations are required to solve the pressure subsystem with ICCG(0). In PMM and the Schur method, we use an approximation of the Schur complement matrix that has information about the variation in viscosity. LSC and MSIMPLER are different from the other preconditioners because the scaling used in both preconditioners do not

have information about the viscosity variation, as the diagonal of the velocity mass matrix is used for scaling the subfactors in the Schur complement matrix. In these experiments we keep the tolerance required for the velocity and the pressure parts the same. Therefore, in Table 8.6, only one tolerance is given for all preconditioners except MSIMPLER, in which 10^{-1} is used for the velocity subsystem and 10^{-3} is used for the pressure subsystem.

Table 8.6: The variable viscosity Stokes problem with Q2-Q1 discretization with AMG/CG for the velocity subsystem and ICCG(0) (PMM, Schur method) or AMG/CG (LSC, LSC_D, MSIMPLER) for the Schur subsystem. Solution accuracy is 10^{-6}.

Grid ↓	Levels/N	PMM	LSC	LSC_D	MSIMPLER	Schur
tol ⟶		10^{-3}	10^{-3}	10^{-6}	10^{-1}, 10^{-3}	10^{-6}
	Iter. (time in seconds)					
66k	3/394	19(51)	11(35)	74(357)	15(35)	1(104)
195k	4/152	18(183)	13(188)	129(2650)	19(138)	1(370)
390k	5/300	18(429)	14(480)	> 1000	19(360)	1(869)
595k	5/408	19(743)	15(871)	> 1000	19(693)	1(1478)
843k	6/112	19(1229)	15(1406)	> 1000	21(989)	1(2686)

PMM and MSIMPLER show better performance than the other iterative methods. In terms of outer iterations, LSC requires fewer outer iterations than PMM and MSIMPLER, but one iteration of LSC is more expensive than one of PMM and MSIMPLER. Figure 8.8 reveals that LSC and MSIMPLER require more pressure iterations than PMM due to the two Poisson solves required. Moreover, the pressure mass matrix approximation itself is also responsible for reduction in iterations for pressure part in PMM. MSIMPLER seems efficient in terms of velocity iterations. The number of AMG/CG iterations taken by PMM and LSC for solving the velocity subsystem, are however comparable. Figure 8.9(a) shows that LSC requires more inner iterations for the velocity subsystem per outer iteration than PMM and MSIMPLER. PMM scales well with the problem size, requires fewer inner iterations for the velocity and pressure subsystems, and, so, it is a better choice than other schemes. The Schur method seems to be two times more expensive than PMM, because it uses an inner accuracy two times greater than PMM. MSIMPLER is efficient than the other schemes, but for larger grids, PMM and the Schur method become better due to their scalibility with problem size.

Figure 8.9(b) shows convergence patterns for different subsystem accuracies in the Schur method. The horizontal line shows the norm of the termination residual. The expected number of iterations for Schur(6), Schur(3) and Schur(2) are 1, 2 and 3, respectively. However, Figure 8.9(b) shows that for lower inner accuracies, the number of iterations is greater than expected. The main reason is that sometimes subsystems accuracies do not guarantee the satisfaction of the outer convergence criteria. In fact,

in some cases, the number of outer iterations required for Schur(6) is greater than 1. The main reason for this is that the accuracy of the update $u_\delta = u_f - u_l$ computed in Step 3 of Algorithm 8.1 might be less accurate than u_f and u_l due to subtraction. As a consequence the overall termination criterion is not satisfied. The same can be the case for steps 5 and 6 in the Schur method. In those cases, we have increased the accuracy of the inner solver a little bit (doubled), which makes the Schur method more expensive.

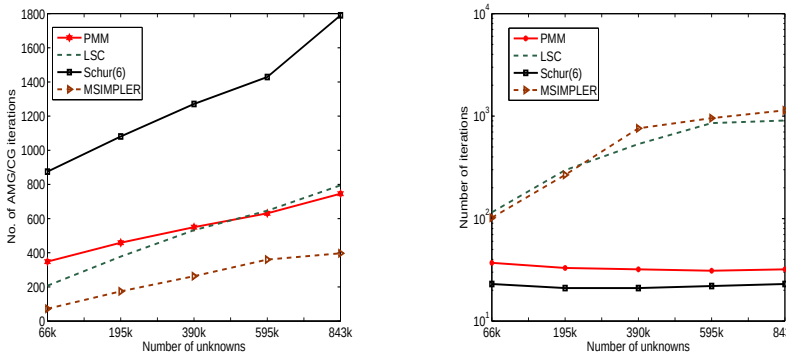
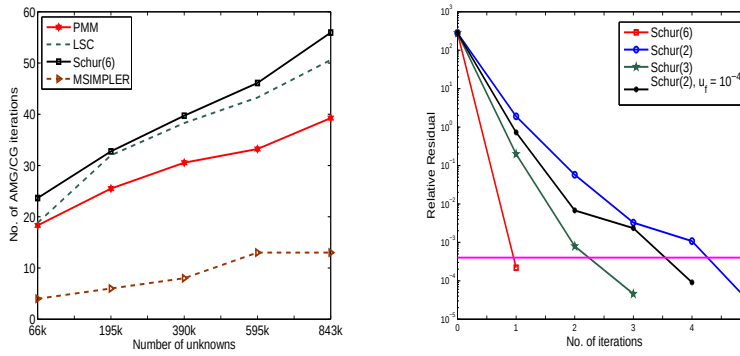


Figure 8.8: Solving variable viscosity Stokes problem with accuracy 10^{-6}: (Same as Figure 8.6).



(a) Same as Figure 8.7.

(b) Solution of the Stokes problem with the Schur method.

Figure 8.9: Extrusion problem results

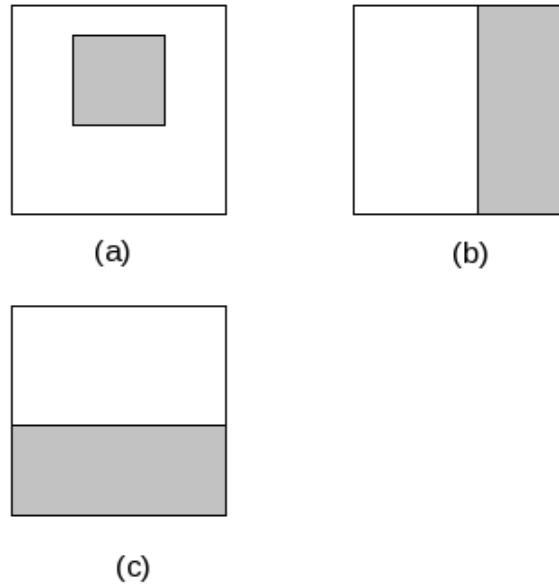


Figure 8.10: Geodynamic problem configurations where the dark region consists of viscosity ν_2 and density ρ_2 and white region has viscosity ν_1 and density ρ_1.

8.3.3 Geodynamic problem having sharp viscosity contrast

The SINKER problem (Figure 8.10(a)) that we consider in this section has already been discussed in Section 8.2. Here we consider a forcing term $f = (0, -\rho g)$, where $g = 9.8m/s^2$. Figure 8.10 shows the 3 configurations that were used to check the convergence and accuracy of the iterative methods. The corresponding problems are referred to as (a), (b), (c). Since LSC and MSIMPLER either diverges or converges very slowly for the high viscosity contrasts considered here, we do not report any results of these methods. The problem and subproblems are scaled as described in Section 8.2, before applying the iterative methods. The velocity solution for all preconditioners is accurate. Therefore we only discuss the accuracy of the pressure solution. This is done by comparing with the "exact solution", computed by a direct method (Error mentioned in tables).

Table 8.7 shows the number of iterations and the norm of the error in pressure for configuration (a). In the first experiments, ν_1 is kept fixed at one and ν_2 is increased; thereafter ν_2 is kept equal to one and ν_1 is increased. We see that the number of iterations for PMM and Schur is almost the same for both the 30×30 and 60×60 grids, which suggests h-independent convergence. LSC_D shows a clear increase with the increase of grid points. For constant ν_2, the difference in accuracy between all

Table 8.7: Iterative solution of the Stokes problem with configuration (a), accuracy = 10^{-6}.

ν	PMM		LSC_D		Schur	
	Iter.	Error	Iter.	Error	Iter. (inner)	Error
30×30						
$\nu_1 = 1,\ \nu_2 = 10^6$	12	9×10^{-4}	26	7×10^{-6}	2(18)	2×10^{-8}
$\nu_1 = 1,\ \nu_2 = 10^3$	12	2×10^{-5}	26	3×10^{-6}	2(20)	2×10^{-10}
$\nu_1 = 1,\ \nu_2 = 10^1$	11	5×10^{-6}	24	1×10^{-6}	2(16)	2×10^{-10}
$\nu_1 = 1,\ \nu_2 = 1$	11	4×10^{-7}	25	2×10^{-6}	2(5)	5×10^{-11}
$\nu_1 = 10^1,\ \nu_2 = 1$	15	1×10^{-6}	27	2×10^{-6}	1(14)	2×10^{-6}
$\nu_1 = 10^3,\ \nu_2 = 1$	18	4×10^{-6}	26	3×10^{-6}	1(18)	2×10^{-6}
$\nu_1 = 10^6,\ \nu_2 = 1$	15	4×10^{-4}	23	1×10^{-4}	1(16)	2×10^{-5}
60×60						
$\nu_1 = 1,\ \nu_2 = 10^6$	13	8×10^{-3}	40	6×10^{-5}	2(19)	5×10^{-8}
$\nu_1 = 1,\ \nu_2 = 10^3$	13	3×10^{-5}	40	5×10^{-6}	2(20)	3×10^{-9}
$\nu_1 = 1,\ \nu_2 = 10^1$	13	1×10^{-6}	41	3×10^{-6}	2(18)	4×10^{-10}
$\nu_1 = 1,\ \nu_2 = 1$	3	6×10^{-6}	36	3×10^{-6}	2(5)	9×10^{-10}
$\nu_1 = 10^1,\ \nu_2 = 1$	16	2×10^{-6}	41	4×10^{-6}	1(14)	4×10^{-6}
$\nu_1 = 10^3,\ \nu_2 = 1$	20	1×10^{-5}	38	7×10^{-6}	1(20)	5×10^{-6}
$\nu_1 = 10^6,\ \nu_2 = 1$	17	4×10^{-4}	35	1×10^{-4}	1(18)	3×10^{-5}

methods is small.

However, in the problem where ν_1 is constant (SINKER), the accuracy obtained with PMM is less than the other two iterative methods, even though all subsystems are solved with high accuracy (10^{-6} or 10^{-7}). The Schur method gives much more accurate results than the other two preconditioners, because the first iteration of the Schur method gives an accurate inner solve, while the second iteration makes the solution more accurate than the desired tolerance. From the table we see, that with respect to accuracy and efficiency, the Schur method seems a better option than the other two. The reason is that PMM requires more iterations than Schur to get the same accuracy, while the costs per iteration are comparable. Similar results have been observed for a 90×90 grid.

Table 8.8 gives the results for problem (b). In principle, all methods reach the same accuracy, but since the Schur method takes 2 outer iterations due to the fixed inner accuracy of 10^{-6}, its final accuracy is better.

Results for problem (c) with a 60×60 grid are given in Table 8.9. We consider the convergence as function of the inner accuracy for two values of ν_2 (10^3 and 10^6). As we already know, the number of outer iterations in these methods depends on the choice of the inner accuracy. One thing that is different from problem (a) is, that, in PMM and the Schur method, inner accuracy also affects the accuracy of the solution of the complete system, while, in LSC_D, the system accuracy remains the same with these changes in the subsystem accuracy. The reason is that problem (c) is relatively

Table 8.8: Iterative solution of the Stokes problem with configuration (b), accuracy = 10^{-6}.

ν	PMM		LSC_D		Schur	
	Iter. (inner)	Error	Iter. (inner)	Error	Iter. (inner)	Error
$60 \times 60,\ \nu_1 = 1$						
$\nu_2 = 10^6$	11	2×10^{-4}	40	2×10^{-4}	2(15)	2×10^{-8}
$\nu_2 = 10^3$	11	1×10^{-5}	40	1×10^{-5}	2(18)	9×10^{-9}
$\nu_2 = 10^1$	11	2×10^{-6}	41	5×10^{-6}	2(16)	2×10^{-10}

simple, because the heavy layer is at the bottom and effect of this layer (buoyancy) is much less than that in problems (a) and (b). This can be seen from the isobars in Figure 8.11. The pressure is smooth for problem (c), but it contains wiggles for both problems (a) and (b).

In this case, the preconditioner itself is the best approximation of the problem, so increasing the inner accuracy has a large effect on the number of outer iterations required and the outer accuracy achieved. In the Schur method, we see that the accuracy obtained for solves with subsystem accuracy 10^{-3} and 10^{-8} is approximately the same. The 3 iterations of the Schur method with inner accuracy 10^{-3} makes the overall solution have roughly the same accuracy as one iteration of the Schur method with inner tolerance 10^{-8}. In this type of problem, PMM seems to be the better option, as we can get an accurate solution by increasing subsystem accuracy.

From all of these experiments, it is clear that to get more accurate results the Schur method is a better option than PMM and LSC. This property can be efficiently utilized in the solution of problems of type (a). A reason that the Schur method gives more accurate results might be the fact that it uses subfactors of the system iteratively in a classical way. If that is the case, we may also expect better results when using Richardson type iterative improvement of the form:

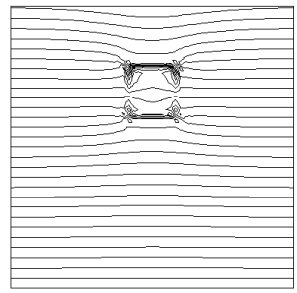
$$x_{k+1} = x_k + PMM^{-1}(b - \mathcal{A}x_k). \tag{8.15}$$

We observe that (8.15) also gives better accuracy than PMM and LSC. Since we need more iterations to apply Richardson, our conclusion regarding using the Schur method in problems of type (a) remains valid.

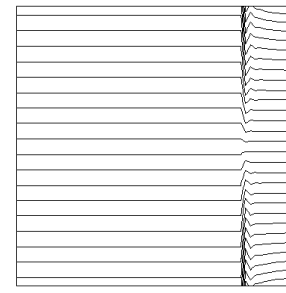
We know that for high viscosity contrast problems, the condition number of the matrix increases with the contrast between the matrix entries. For a 10^8 contrast, the condition number of the matrix is expected to be of the same order or higher even on the small meshes. Solving the problem with relative termination tolerance of 10^{-6} results in an error norm of roughly 10^2 (contrast·tolerance). Scaling does not change the condition number of the system (its only affects the termination criterion). So, in order to get the same order of accuracy for the non-scaled problem, we need a termination criterion that is more severe than the previous 10^{-6}. In this scenario, the use of the Schur method can be a better option than other type of iterative methods.

Table 8.9: Iterative solution of the Stokes problem with configuration (c), accuracy = 10^{-6}.

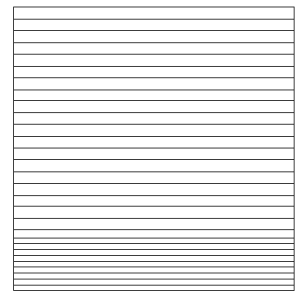
inner acc.	PMM		LSC_D		Schur	
	Iter. (inner)	Error	Iter. (inner)	Error	Iter. (inner)	Error
$\nu_1 = 1,\ \nu_2 = 10^6$						
10^{-3}	9	5×10^{-4}	39	4×10^{-4}	3(10)	5×10^{-6}
10^{-6}	9	7×10^{-4}	35	3×10^{-4}	2(10)	3×10^{-9}
10^{-8}	2	7×10^{-6}	33	2×10^{-4}	1(1)	7×10^{-6}
Direct	2	1×10^{-8}	31	2×10^{-4}	1(1)	9×10^{-9}
$\nu_1 = 1,\ \nu_2 = 10^3$						
10^{-3}	11	1×10^{-5}	41	2×10^{-5}	3(11)	4×10^{-7}
10^{-6}	9	4×10^{-5}	36	8×10^{-6}	2(10)	3×10^{-10}
10^{-8}	2	8×10^{-7}	34	9×10^{-6}	1(1)	8×10^{-7}
Direct	2	2×10^{-8}	32	9×10^{-6}	1(1)	9×10^{-9}



(a) Problem (a), $\nu_1 = 1,\ \nu_2 = 10^6$



(b) Problem (b), $\nu_1 = 1,\ \nu_2 = 10^6$



(c) Problem (c), $\nu_1 = 1,\ \nu_2 = 10^6$

Figure 8.11: The pressure solution in various configurations.

8.4 Summary

In this chapter, we consider the solution of the incompressible Stokes problem using preconditioned iterative methods. We solve three different classes of problems with various configurations and viscosity distributions using PMM, LSC, MSIMPLER and the Schur method. For the isoviscous problem, PMM and the Schur method show better performance than the other preconditioners. We note that for this type of problem based on a FEM discretization on an unstructured grid, the combination of PMM using AMG/CG to solve the subsystems leads to a number of inner and outer iterations that is essentially independent of h. For the variable viscosity problem, the performance of PMM and the Schur method come close to one another due to the high accuracy requirements for the subsystem solves in PMM. For the Schur method used as direct method, the accuracy of the inner subsystems must be kept equal to the outer accuracy. In general, this strategy works for isoviscous problems. However, in a variable viscosity problem, the Schur method often needs a second iteration. This can be avoided by solving the last velocity subsystem with a higher accuracy, in some cases, all subsystems should be solved using a higher accuracy than the desired tolerance for the complete system. For a certain range of problems, MSIMPLER and LSC perform better than PMM and the Schur method. However due to its h-dependent convergence, its performance becomes equal to or worse than that of PMM and the Schur method for large size problems. For a large viscosity contrast, LSC and MSIMPLER fail to converge. This suggests that for a high viscosity contrast problem, the Schur complement matrix must be approximated by an operator that contains viscosity information for the problem. The complete system is also required to be scaled with proper scaling factors. The Schur method is a good choice to use in high viscosity contrast problems because it gives more accurate solution at lesser cost than PMM and LSC_D.

Chapter 9

Conclusions and future research

9.1 Conclusions

This thesis is devoted to preconditioners for the incompressible Navier-Stokes equations. The final goal is to develop solvers that are both fast and robust. However, it is hard to find solvers that satisfy both properties, because solvers that are robust are usually not optimal. For the Navier-Stokes equations, the ultimate goal is to develop a preconditioner that gives mesh and Reynolds number independent convergence. Preconditioned Krylov methods or multigrid techniques are considered to be the best choices to solve large linear systems. In this thesis, we focus only on preconditioners to accelerate a Krylov methods. Multigrid is used to solve subsystems only.

In Chapter 4 we gave an overview of the most popular preconditioners in the literature. Based on some experiments and its convergence behavior we choose LSC for comparison with the schemes we developed ourselves. Besides that some of our improved techniques are based on LSC.

Due to its ease of implementation and applicability, algebraic preconditioners are always attractive. However, since there is no knowledge of the complete system involved in construction they may suffer from bad convergence. The common approach to make them efficient in case of indefinite systems is to apply pivoting or alternatively renumbering of nodal points. In Chapter 5, we presented a new reordering technique that is simple but effective and uses extra information of the discretized equations. First the grid points are renumbered with a suitable renumbering scheme. The renumbering scheme can be any one that gives an optimal nonzero profile. In our case, we use the Sloan and Cuthill-McKee renumbering schemes. After grid renumbering, the unknowns are reordered per level. In such renumbering, unknowns are intermixed with each other such that it seems that the matrix consists of a sequence of smaller subsystems. The resulting method is called the SILU preconditioner. From our experiments it is clear that the *p-last-per level* technique gives better convergence than the other schemes discussed. In 2D problems, Sloan renumbering with p-last per level

reordering leads to the best results both for Taylor-Hood and Crouzeix-Raviart elements. In 3D problems, Cuthill-McKee renumbering gives fast convergence for the Q2-Q1 discretization, whereas for the Q2-P1 discretization, Sloan renumbering gives a better convergence. The convergence of the SILU preconditioner shows a dependence on the grid size and mild dependence on the Reynolds number.

In Chapter 6 SIMPLE-type preconditioners are discussed based on the classic SIMPLE method of Patankar. The convergence of the classic method depends on relaxation parameters which we have to be chosen by trial and error. An alternative is to use SIMPLE as preconditioner. Since only one SIMPLE step per iteration is used there is no need for relaxation. Experiments showed that relaxation may improve convergence in case of Stokes a little bit, but there is no effect at all for Navier-Stokes. The convergence rate of SIMPLE increases with increasing in grid size. The alternative is to use SIMPLER which is more expensive per iterations but gives faster convergence. Unfortunately its performance depends non-uniformly on the inner accuracies, which makes this choice unreliable. In case of Stokes SIMPLER can be improved by starting with SIMPLE step (hSIMPLER), but that does not solve the inner accuracy dependence.

The above variants of SIMPLE use the diagonal of the velocity matrix in scaling the Schur complement and updates. A much better convergence is attained if the diagonal of the velocity mass matrix is used as scaling operator. This approach is called MSIMPLER (Modified SIMPLER). MSIMPLER is superior to the other SIMPLE variants and shows faster convergence. It has only a mild dependence on the grid size and Reynolds number. Its convergence is almost independent of inner accuracies. Therefore, the preconditioner is efficient in both outer/inner iterations and CPU time.

SIMPLE, MSIMPLER, LSC and SILU are compared in 2D and 3D in Chapter 7. MSIMPLER proved to be cheaper than SILU, especially in large problems or when the problem is solved with high accuracy. The number of outer iterations in MSIMPLER hardly increases if a direct solver for the subsystems is replaced by an iterative solver. This is in contrast with LSC where large differences are observed. It appears that the combination of LSC with MG is almost optimal. The combination of LSC with a PCG inner solver can take many iterations and much CPU time. MSIMPLER performs better with both types of inner solvers. Convergence characteristics of LSC and MSIMPLER are almost the same, they both show mild dependence on Reynolds number and mesh size.

We compared IDR(s) and Bi-CGSTAB(ℓ) both preconditioned with SILU. For equidistant grids, performance of both Krylov method is comparable. However, when the grid is stretched and the preconditioner becomes less efficient, IDR(s) starts to perform better than Bi-CGSTAB(ℓ) for optimal choice of s and ℓ.

The previous discussion is based on solving the Navier-Stokes equations with constant viscosity. For a number of physical processes variable viscosity models are important. In Chapter 8, we present preconditioners that can be effective to solve the variable viscosity Stokes problem. The new methods are based on the pressure mass matrix approximation of the Schur complement matrix. In order to make pressure mass matrix spectrally equivalent to the Schur complement matrix in case of constant

viscosity, it is multiplied by the inverse of the viscosity. In case of variable viscosity the best approach is to scale the pressure mass matrix at element level while creating this matrix. Since AMG performs better on scalar fields, it would be better if we have constant degree of freedom at each grid point. Therefore, it is necessary to treat Dirichlet boundary conditions as a penalized slip boundary condition. To get convergence in MSIMPLER and LSC it is necessary to update the velocity mass matrix by multiplying the entries of the boundary elements with the approximated boundary conditions by the penalty parameter. The reason is that the velocity matrix contains the same parameter in the corresponding rows.

It has been shown that for the right preconditioned GCR we use, the stopping criterion needs to be adapted in order to get accurate results. Especially in case PMM (see Section 8.1.1) is used as preconditioner, GCR may terminate too soon, giving an inaccurate solution. The stopping criterion is corrected by scaling the complete system or by scaling the residual with a scaling matrix. The alternative is to base the stopping criteria on the preconditioned residual.

In all Stokes examples we investigated, varying from constant viscosity to sharp viscosity contrasts the block triangular preconditioner PMM and the Schur method, using the pressure mass matrix as preconditioner outperform the classical preconditioners. Also MSIMPLER shows better convergence and for extrusion problem, it appears to be the favorite. However, due to mild dependence on mesh size we expect that for very fine grids PMM will be the best choice. In case of large viscosity contrasts both LSC and MSIMPLER fail to converge. Probably this is because the preconditioner does not contain viscosity information. LSC_D solves this problem for LSC, but from our experiments it seems that the Schur method is to be preferred since it gives more accurate solution at lesser cost than PMM and LSC_D.

9.2 Ideas for future research

- The saddle point ILU preconditioner (SILU) performs well in many cases. SILU is based on the ILU(0) concept. Making ILU(0) efficient for an indefinite system, is always a difficult task. The performance of the preconditioner might be improved by better renumbering and reordering algorithms. A better selection of levels might also increase the efficiency especially in 3D. An alternative is to use extra fill-in which requires extra memory but improves the quality of the preconditioner.

- It is clear from our experiments that the key of success for the block preconditioners, is the ability to approximate the Schur complement matrix. An important part of the matrix $BF^{-1}B^T$ is the approximation of F^{-1}. For example MSIMPLER performs better than SIMPLE due to a better approximation of F^{-1}. In the future more work can be done to find even better approximations.

- In problems, with large jumps in viscosity, we have small eigenvalues $O(\nu^{-1})$ in the region having large viscosity. Large viscosity jumps give rise to eigenvalues

close to zero causing stagnation of the Krylov solver. To improve convergence, deflation type schemes can be developed to get rid of these bad eigenvalues. Application of deflation type schemes on saddle point problem can be an attractive area of further research.

- The convergence of all preconditioners for Stokes and Navier-Stokes treated in this thesis is effected by stretching. Elongated domains have the largest effect on the convergence especially in case of a pressure mass matrix. The reason is, that condition number of the mass matrix increases with the increase in grid size if grid does not have quasi uniform subdivision. Although we have reached an improvement in this field, much more work must be done in finding the cause and solution of bad convergence for stretched problems.

- To reduce the wall-clock time of the computations, parallelization of the solvers should be investigated.

Appendix A

Grid reordering schemes

A.1 Sloan renumbering scheme

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.

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

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 $w = \max_{i \leq j \leq k} |L_j(i)| < w_{min}$, 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.

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 $n_{max} = \max_{1 \leq i \leq N_t} n_i$. For convenience n_{max} may be set equal to N_t (since the maximum current degree in any graph with N_t nodes is N_t). 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 A.1 Cuthill and McKee's Algorithm

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

Bibliography

[1] M. A. Ajiz and A. Jennings. A robust incomplete Choleski-conjugate gradient algorithm. *International Journal for Numerical Methods in Engineering*, 20(5):949–966, 1984.

[2] M. Arioli and D. Loghin. Stopping criteria for mixed finite element problems. *ETNA*, 29:178–192, 2008.

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

[4] K. J. Arrow, L. Hurwicz, and H. Uzawa. *Studies in Linear and Non-Linear Programming*. Stanford University Press, Stanford, 1958.

[5] O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, Cambridge, 1994.

[6] I. Babuška. The Finite element method with Lagrange multipliers. *Numer. Math.*, 20:179–192, 1973.

[7] M. Benzi. Preconditioning Techniques for Large Linear Systems: A Survey. *J. Comput. Phys.*, 182(2):418–477, 2002.

[8] M. Benzi, H. Choi, and D. Szyld. Threshold Ordering for Preconditioning Nonsymmetric Problems. In *G. Golub et al., editors, Sci. Comput., Proc. Workshop. Hong Kong*, pages 159–165. Springer Verlag, 10–12 March 1997.

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

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

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

[12] M. Benzi and M. Tuma. A Robust Incomplete Factorization Preconditioner for Positive Definite Matrices. *Numerical Linear Algebra with Applications*, 10:385–400., 2003.

[13] J. Blasco, R. Codina, and A. Huerta. A fractional-step method for the incompressible Navier-Stokes equations related to a predictor-multicorrector algorithm. *International Journal for Numerical Methods in Fluids*, 28(10):1391–1419, 1998.

[14] M. Bollhöfer and Y. Saad. Multilevel Preconditioners Constructed From Inverse-Based ILUs. *SIAM J. Sci. Comput.*, 27(5):1627–1650, 2006.

[15] D. Braess and R. Sarazin. An efficient smoother for the Stokes problem. *Appl. Numer. Math.*, 23(1):3–19, 1997.

[16] A. Brandt, S. F. McCormick, and J. W. Ruge. *Algebraic multigrid (AMG) for sparse matrix equations*. Cambridge University Press, Cambridge, 1984.

[17] F. Brezzi. On the Existence, Uniqueness Approximation of saddle-point problems arising from Lagrange multipliers. *RAIRO, série rouge, Analyse Numérique R-2*, pages 129–151, 1974.

[18] W. L. Briggs, V. E. Henson, and S. F. McCormick. *A multigrid tutorial: second edition*. Society for Industrial and Applied Mathematics, 2000.

[19] C. Burstedde, O. Ghattas, M. Gurnis, G. Stadler, E. Tan, T. Tu, L. C. Wilcox, and S. Zhong. Scalable adaptive mantle convection simulation on petascale supercomputers. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, pages 1–15, Austin, Texas, 2008. IEEE Press.

[20] A. J. Chorin. Numerical Solution of the Navier-Stokes Equations. *Mathematics of Computation*, 22(104):745–762, 1968.

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

[22] M. Crouzeix and P. A. Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations. *Rairo Analyse Numerique*, 7:33–76, 1973.

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

[24] C. Cuvelier, A. Segal, and A. A. van Steenhoven. *Finite element methods and Navier-Stokes equations*. Reidel Publishing Company, Dordrecht, Holland, 1986.

[25] O. Dahl and S. Ø. Wille. An ILU preconditioner with coupled node fill-in for iterative solution of the mixed finite element formulation of the 2D and 3D Navier-Stokes equations. *Int. J. Numer. Meth. Fluids*, 15(5):525–544, 1992.

[26] A. C. de Niet and F. W. Wubs. Two preconditioners for saddle point problems in fluid flows. *Int. J. Numer. Meth. Fluids*, 54(4):355–377, 2007.

[27] I. Duff, I. Erisman, and J. Reid. *Direct methods for sparse matrices*. Oxford University Press, London, England, 1986.

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

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

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

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

[32] H. Elman, V.E. Howle, J. Shadid, R. Shuttleworth, and R. Tuminaro. A taxonomy and comparison of parallel block multi-level preconditioners for the incompressible Navier-Stokes equations. *Journal of Computational Physics*, 227(3):1790–1808, 2008.

[33] H. C. Elman. Preconditioning for the Steady-State Navier–Stokes Equations with Low Viscosity. *SIAM Journal on Scientific Computing*, 20(4):1299–1316, 1999.

[34] H. C. Elman. Preconditioning Strategies for Models of Incompressible Flow. *J. Sci. Comput.*, 25(1):347–366, 2005.

[35] H. C. Elman, D. Silvester, and A. J. Wathen. *Finite Elements and Fast Iterative Solvers with applications in incompressible fluids dynamics*. Oxford University Press, Oxford, 2005.

[36] R. Fletcher. Conjugate gradient methods for indefinite systems. pages 73–89. 1976.

[37] M. Fortin. Old and new finite elements for incompressible flows. *Int. J. Numer. Meth. Fluids*, 1(4):347–364, 1981.

[38] A. Gauthier, F. Saleri, and A. Veneziani. A fast preconditioner for the incompressible Navier Stokes Equations. *Comput. Vis. Sci.*, 6(2):105–112, 2004.

[39] T. Geenen, M. ur Rehman, S. P. MacLachlan, G. Segal, C. Vuik, A. P. van den Berg, and W. Spakman. Scalable robust solvers for unstructured FE geodynamic modeling applications: Solving the Stokes equation for models with large localized viscosity contrasts. *Geochem. Geophys. Geosyst.*, 10(9):–, 2009.

[40] G. Golub and C. Van Loan. *Matrix computations*. John Hopkins University Press, Baltimore, MD, 1996.

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

[42] A. Janka. Smoothed aggregation multigrid for a Stokes problem. *Computing and Visualization in Science*, 11(3):169–180, 2008.

[43] D. Kay, D. Loghin, and A. Wathen. A Preconditioner for the Steady-State Navier-Stokes Equations. *SIAM J. Sci. Comput.*, 24(1):237–256, 2002.

[44] H. P. Langtangen. Conjugate gradient methods and ILU preconditioning of non-symmetric matrix systems with arbitrary sparsity patterns. *International Journal for Numerical Methods in Fluids*, 9(2):213–233, 1989.

[45] C. Li and C. Vuik. Eigenvalue analysis of the SIMPLE preconditioning for incompressible flow. *Numer. Lin. Alg. Appl.*, 11(5-6):511–523, 2004.

[46] M. Sala C. Siefert M. Gee, J. Hu and R. Tuminaro. *ML 5.0 Smoothed Aggregation User's Guide*, sandia report sand2006-2649 edition.

[47] M. Manguoglu, A. Sameh, T. Tezduyar, and S. Sathe. A nested iterative scheme for computation of incompressible flows in long domains. *Computational Mechanics*, 43(1):73–80, 2008.

[48] M. Manguoglu, A. H. Sameh, F. Saied, T. E. Tezduyar, and S. Sathe. Preconditioning Techniques for Nonsymmetric Linear Systems in the Computation of Incompressible Flows. *Journal of Applied Mechanics*, 76(2):021204, 2009.

[49] T.A. Manteuffel. An incomplete factorization technique for positive definite linear systems. *Math. Comput.*, 34:73–497, 1980.

[50] D. A. May and L. Moresi. Preconditioned iterative methods for Stokes flow problems arising in computational geodynamics. *Physics of the Earth and Planetary Interiors*, 171(1-4):33–47, 2008.

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

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

[53] G. Meurant. *Computer solution of large linear systems*, volume Vol. 28 of *Studies in Mathematics and Its Applications*. Elsevier, Amsterdam, 1999.

[54] N. Munksgaard. Solving Sparse Symmetric Sets of Linear Equations by Preconditioned Conjugate Gradients. *ACM Trans. Math. Softw.*, 6(2):206–219, 1980.

[55] M. F. Murphy, G. H. Golub, and A. J. Wathen. A Note on Preconditioning for Indefinite Linear Systems. *SIAM J. Sci. Comput.*, 21(6):1969–1972, 2000.

[56] Y. Notay. Flexible Conjugate Gradients. *SIAM J. Sci. Comput.*, 22(4):1444–1460, 2000.

[57] M. Olshanskii and P. Grinevich. An iterative method for the Stokes type problem with variable viscosity. *SAIM J. Sci. Comput.*, Submitted, 2008.

[58] M. A. Olshanskii. An Iterative Solver for the Oseen Problem and Numerical Solution of Incompressible Navier-Stokes Equations. *Numer. Linear Algebra Appl.*, 6:353–378, 1999.

[59] M. A. Olshanskii and Y. V. Vassilevski. Pressure Schur Complement Preconditioners for the Discrete Oseen Problem. *SIAM Journal on Scientific Computing*, 29(6):2686–2704, 2007.

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

[61] J. B. Perot. An analysis of the fractional step method. *J. Comput. Phys.*, 108(1):51–58, 1993.

[62] J. W. Ruge and K. Stüben. Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). In D. J. Paddon and H. Holstein, editors, *Multigrid Methods for Integral and Differential Equations*, The Institute of Mathematics and its Applications Conference Series, pages 169–212. Clarendon Press, Oxford, 1985.

[63] J. W. Ruge and K. Stüben. Algebraic Multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods, Frontiers in Applied Mathematics*, volume 3, pages 73–130. SIAM, Philadelphia, 1987.

[64] Y. Saad. *Iterative Methods for Sparse Linear Systems*. Society for Industrial and Applied Mathematics, Philadelphia, 2003.

[65] Y. Saad. Multilevel ILU With Reorderings for Diagonal Dominance. *SIAM J. Sci. Comput.*, 27(3):1032–1057, 2005.

[66] Y. Saad and M. H. Schultz. GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 7(3):856–869, 1986.

[67] Y. Saad and H. A. van der Vorst. Iterative solution of linear systems in the 20th century. *J. Comput. Appl. Math.*, 123(1-2):1–33, 2000.

[68] G. Segal and C. Vuik. A simple iterative linear solver for the 3D incompressible Navier-Stokes equations discretized by the finite element method. Technical Report DUT-TWI-95-64, Delft, The Netherlands, 1995.

[69] D. Silvester, H. Elman, D. Kay, and A. Wathen. Efficient preconditioning of the linearized Navier-Stokes equations for incompressible flow. *J. Comput. Appl. Math.*, 128(1-2):261–279, 2001.

[70] D. Silvester and A. Wathen. Fast iterative solution of stabilised Stokes systems part II: using general block preconditioners. *SIAM J. Numer. Anal.*, 31(5):1352–1367, 1994.

[71] H.D. Simon. Incomplete LU preconditioners for conjugate gradient type iterative methods. In *In Proceedings of the Eighth SPE symposium on Reservoir Simulations*, number Paper SPE 13533, 1985.

[72] G.L.G. Sleijpen and D.R. Fokkema. BiCGstab(ell) for Linear Equations involving Unsymmetric Matrices with Complex Spectrum ,. *ETNA*, 1:11–32, 1993.

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

[74] P. Sonneveld. CGS, A Fast Lanczos-Type Solver for Nonsymmetric Linear systems. *SIAM J. Sci. and Stat. Comput.*, 10(1):36–52, 1989.

[75] P. Sonneveld and M. B. van Gijzen. IDR(s): a family of simple and fast algorithms for solving large nonsymmetric linear systems. *SIAM J. Sci. Comput.*, 31(2):1035–1062, 2008.

[76] K. Stüben. A review of algebraic multigrid. *J. Comput. Appl. Math.*, 128(1-2):281–309, 2001.

[77] C. Taylor and P. Hood. A numerical solution of the Navier-Stokes equations using the finite element techniques. *Computers and Fluids*, 1:73–100, 1973.

[78] U. Trottenberg, C.W. Oosterlee, and A. Schüller. *Multigrid*. Number SBN 0-12-701070-X. Academic Press, London, 2001.

[79] M. ur Rehman, C. Vuik, and G. Segal. Preconditioners for the incompressible Navier-Stokes equations. Report 07-15, Delft University of Technology, Delft Institute of Applied Mathematics, Delft, 2007.

[80] A. van der Sluis. Condition numbers and Equilibration of Matrices. *Numer. Math.*, 14:14–23, 1969.

[81] A Van der Sluis and H.A Van der Vorst. The rate of convergence of conjugate gradients. *Numer.Math.*, 48:543–560, 1986.

[82] H. A. van der Vorst. Bi-CGSTAB: A Fast and Smoothly Converging Variant of Bi-CG for the Solution of Nonsymmetric Linear Systems. *SIAM J. Sci. Stat. Comput.*, 13(2):631–644, 1992.

[83] H. A. van der Vorst. *Iterative Krylov Methods for Large Linear systems*. Cambridge University Press, Cambridge, 2003.

[84] H. A. van der Vorst and C. Vuik. GMRESR: a family of nested GMRES methods. *J. Numer. Lin. Alg. Appl.*, 1(4):369–386, 1994.

[85] H.A. van der Vorst and C. Vuik. The superlinear convergence behaviour of GMRES. *J. Comp. Appl. Math.*, 48:327–341, 1993.

[86] P. Vaněk, J. Mandel, and M. Brezina. Algebraic multigrid by smoothed aggregation for second and fourth order elliptic problems. *Computing*, 56(3):179–196, 1996.

[87] R. S. Varga. *Matrix Iterative Analysis*. Prentice-Hall, Englewood Cliffs, New Jersey, 1962.

[88] C. Vuik and A. Saghir. The Krylov accelerated SIMPLE(R) method for incompressible flow. Report 02-01, Delft University of Technology, Department of Applied Mathematical Analysis, Delft, 2002.

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

[90] C. Vuik, A. Segal, J.A. Meijerink, and G.T. Wijma. The construction of projection vectors for a Deflated ICCG method applied to problems with extreme contrasts in the coefficients. *Journal of Computational Physics*, 172:426–450, 2001.

[91] A. Wathen. Preconditioning and convergence in the right norm. *Int. J. Comput. Math.*, 84(8):1199–1209, 2007.

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

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

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

Publications

Journal publications

- M. ur Rehman and C. Vuik and G. Segal. *A comparison of preconditioners for incompressible Navier-Stokes solvers*, International Journal for Numerical Methods in Fluids. Vol. 57, pp. 1731-1751, 2008.
- M. ur Rehman and C. Vuik and G. Segal. *Preconditioners for the Steady Incompressible Navier-Stokes Problem* , IAENG International Journal of Applied Mathematics. Vol. 38, No. 4, pp. 223-232, 2008.
- M. ur Rehman and C. Vuik and G. Segal. *SIMPLE-type preconditioners for the Oseen problems*, International Journal for Numerical Methods in Fluids. Vol. 61 No. 4, pp. 432-452, 2009.
- T. Geenen, M. ur Rehman, S.P. MacLachlan, G. Segal, C. Vuik, A. P. van den Berg, and W. Spakman. *Scalable robust solvers for unstructured FE geodynamic modeling applications; solving the Stokes equation for models with large, localized, viscosity contrasts* , Geochemistry, Geophysics, Geosystems.Vol.10, 2009.
- M. ur Rehman and T. Geenen and C. Vuik and G. Segal and S. P. MacLachlan. *On iterative methods for the incompressible Stokes problem*, International Journal for Numerical Methods in Fluids. Accepted 2009.
- A. Segal, M. ur Rehman, and C. Vuik. *Preconditioners for incompressible Navier-Stokes solvers*, Numerical Mathematics: Theory, Methods and Applications. submitted 2009.

Conference proceedings

- M. ur Rehman and C. Vuik and G. Segal. *Numerical solution techniques for the steady incompressible Navier-Stokes problem*, World Congress on Engineering

2008, London, U.K., 2-4 July, 2008.Volume II Editors S.I. Ao and L. Gelman and D.W.L. Hukins and A. Hunter and A.M. Korsunsky pp. 844–848, International Association of Engineers, Hong Kong, 2008 SBN: 978-988-17012-3-7. Won the Best Student Paper Award.

- M. ur Rehman, C. Vuik, G. Segal. *Block preconditioners for the incompressible Stokes problem,* Seventh International Conference on "Large Scale Scientific Computations", June 4-8, 2009, Sozopol, Bulgaria. To appear in Lecture Notes in Computer Sciences (LNCS).

Talks

- *Preconditioners for the incompressible Navier-Stokes problem.* 3rd International conference on 21st century mathematics, Lahore-Pakistan, March 4-7, 2007.
- *An advanced ILU preconditioner for the incompressible Navier-Stokes problem.* Computational Methods with Applications, Harrachov - Czech Republic, August 19-25, 2007.
- *Numerical Solution Techniques for the Steady Incompressible Navier-Stokes Problem.* The 2008 International Conference of Applied and Engineering Mathematics , London - U.K., July 2-4, 2008.
- *SIMPLE-type preconditioners for the Oseen problem.* Burgersdag TU Eindhoven -The Netherlands, January 13, 2009.
- *Preconditioning techniques for the incompressible Stokes equations.* Seventh International Conference on "Large Scale Scientific Computations", Sozopol-Bulgaria, June 4-8, 2009.
- *Block preconditioners for the incompressible Stokes problem.* ENUMATH Uppsala University-Sweden, June 29 - July 3, 2009.

Curriculum Vitae

Mehfooz ur Rehman was born on 11th October 1973 in Seni Gumbat, Kohat, Pakistan. He did his primary and secondary education from Government High School Seni Gumbat, Kohat in 1989. He did his F.Sc. from P.A.F college Kohat in 1992. From 1992-1997, he attended University of Engineering and Technology Peshawar for his degree in B.Sc. Electrical Engineering. He was awarded fellowship to do his Master degree in Systems Engineering from Pakistan Institute of Engineering and Applied Sciences, Islamabad in 1997. From 1999 -2005, he did job in NESCOM, a research and development organization in Islamabad. In 2005, he was awarded scholarship from Higher Education Commission (HEC), Pakistan to persue his PhD degree. Since 2005, he is PhD student in Numerical Analysis Group at TU Delft under supervision of Prof. Dr. Ir. Kees Vuik and Ir. Guus Segal.

He worked on iterative methods for the incompressible Navier-Stokes equations. This thesis consists of the work that he has been published in various journals and conference contributions. He gave talks in various conferences including Lahore -Pakistan 2007, Harrachov-Czech Republic 2007, London-UK 2008, Eindhoven-Netherlands 2009, Sozopol-Bulgaria 2009 and Uppsala-Sweden 2009.

He enjoys playing Cricket and Table Tennis. His interests includes programming in numerical methods.