Solution of the coupled Navier-Stokes equations

Report 95-28

C. Vuik
A. Segal

TU Delft
Technische Universiteit Delft
Delft University of Technology

Faculteit der Technische Wiskunde en Informatica
Faculty of Technical Mathematics and Informatics

ISSN 0922-5641

Copies of these reports may be obtained from the bureau of the Faculty of Technical Mathematics and Informatics, Julianalaan 132, 2628 BL Delft, phone +3115784568.
A selection of these reports is available in PostScript form at the Faculty's anonymous ftp-site, ftp.twi.tudelft.nl. They are located in directory /pub/publications/tech-reports. They can also be accessed on the World Wide Web at:
http://www.twi.tudelft.nl/TWI/Publications/Overview.html

Solution of the coupled Navier-Stokes equations

C. Vuik and A. Segal
Faculty of Technical Mathematics and Informatics,
Delft University of Technology,
Mekelweg 4, 2628 CD Delft, The Netherlands,
e-mail: c.vuik@math.tudelft.nl a.segal@math.tudelft.nl

Abstract

In this paper the incompressible Navier-Stokes equations are discretized by the finite element method. After linearization large, sparse systems of linear equations have to be solved. A well known problem is the occurrence of zero elements on the main diagonal. We describe ordering techniques of the grid points and the unknowns to avoid this problem, so direct and iterative methods can be used without pivoting. It appears that in three-dimensional problems the iterative methods are much better than the direct or penalty methods.

1 Introduction

There are several methods to solve the incompressible Navier-Stokes equations. In this paper we restrict ourselves to the three-dimensional case and use the velocities and the pressure as unknowns. The coupled equations, three momentum equations and the continuity equation, are discretized by the finite element method. After linearization we obtain a large, sparse system of linear equations. Due to zero elements on the main diagonal, break down of direct and iterative methods may occur.

In Section 2, we describe and investigate some orderings, where the zero main diagonal elements become non zero elements during the decomposition of the matrix. In our numerical experiments we observe no break down of the direct method without pivoting.

Thereafter, in Section 3, we specify two incomplete LU decompositions, which are used as preconditioners for Krylov subspace methods. Some theoretical results are given.

Finally the described methods are compared numerically in Section 4. It appears that already for small problem sizes the preconditioned Krylov subspace methods are much better than the direct or penalty methods. For large problem sizes it is impossible to use direct or penalty methods due to excessive memory requirements. Furthermore for medium problem sizes the CPU time for a direct or penalty method is orders of magnitudes larger than that for the iterative method.

2 Statement of the problem and ordering techniques

In this section we consider the discretization of the incompressible Navier-Stokes equations by finite element methods. Furthermore the consequences of the incompressibility condition are investigated. The general form of the stationary Navier-Stokes equations may be written as

$$-div\ \boldsymbol{\sigma} + \mathbf{u} \cdot \nabla \mathbf{u} = \mathbf{f} \tag{1}$$

where $\boldsymbol{\sigma}$ is the stress tensor:

$$\boldsymbol{\sigma} = -p\mathbf{I} + \mu(\nabla \mathbf{u}^T + \nabla \mathbf{u}) \tag{2}$$

μ denotes the viscosity, $\mathbf{f}$ is some external force and $\mathbf{u} \cdot \nabla \mathbf{u}$ represents the convective terms. The continuity equation is given by

$$div\ \mathbf{u} = 0 \tag{3}$$

Equations (1) to (3) are solved by a standard finite element method based upon the Galerkin formulation as can be found in for example Cuvelier et al [2]. In this paper we restrict ourselves to the three-dimensional case. The elements used are the triquadratic isoparametric Crouzeix Raviart hexahedrons. This means that the velocity is approximated by a quadratic polynomial in each direction in the reference element and the pressure is approximated by a linear polynomial, which is discontinuous over the element boundaries. Since the convective terms are non-linear it is necessary to use some linearization scheme. To that end we use the solution of the Stokes equations (i.e. the Navier-Stokes equations without convective terms) as initial guess, proceed with a number of so-called Picard iterations and finally apply Newton linearization. In each step of the iteration process, it is necessary to solve a large system of linear equations. Formally this system of equations can be written as:

$$\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}, \tag{4}$$

where A denotes the discretization of the stress tensor and the linearized convection terms, $B^T p$ the discretization of the pressure gradient and $Bu = 0$ the discretization of the continuity equation. The vector u represents the velocity unknowns and the vector p the unknown pressures.

The so-called *zero-block* at the main diagonal of the second equation of (4) is caused by the fact that the pressure is not present in the continuity equation, whereas the equation itself is coupled to the pressure unknowns. This property is inherent to incompressible Navier-Stokes equations and is independent of the type of discretization. Due to this *zero-block* a so-called saddle point problem arises. This is one of the major problems in the solution of the incompressible Navier-Stokes problems.

The system of linear equations (4) may be solved by either a direct method (LU-decomposition) or an iterative method. Let us first focus on the direct methods.

The ordering as suggested in equation (4), that is, first all velocity unknowns and then all pressure unknowns, appears to be very uneconomical, since the corresponding profile is very large. In the sequel we shall denote this ordering by p-last ordering. A much better ordering is achieved by ordering all unknowns per point. If a good nodal point numbering is present this gives by far most the smallest profile. Unfortunately the *zero-block* complicates things considerably. Due to boundary conditions it may be possible that the first diagonal element in the matrix is a zero and as a consequence straight-forward LU-decomposition is not possible. In that case it is necessary to use a kind of pivoting strategy, which influences the structure of the matrix and makes the estimation of the size of the matrix a difficult task. So in fact one would like to have some a priori numbering of nodal points and unknowns that prevents the presence of a zero diagonal element during the elimination process and moreover, produces a kind of optimal profile. This numbering must be such that the first unknowns are velocity degrees of freedom independent of the type of boundary conditions. Furthermore it must have a very small influence at the local band width.

Let us, for the sake of the argument, consider the simple rectangular domain of Figure 1, subdivided into triangles. Furthermore we assume that the velocity is prescribed at the complete boundary. If the nodes are numbered in a natural way, from left to right and line-wise from below to the upper boundary, it is clear that one of the first unknowns is the pressure in the first element. The band width is determined by one stroke of elements only. One may expect that if we first number the velocities and then the pressures, then during the elimination procedure the pressure diagonal elements become non-zero. In order to avoid increase of the profile a clever numbering would be to renumber the unknowns such that per stroke of elements first all velocities are numbered and then all pressures. In this way, due to fill in, the elimination may change the pressure diagonal elements to non-zero elements and the actual band width is hardly changed. Such a renumbering will be called pressure last renumbering per level or p-last per level.

This renumbering is simple for a rectangular region. In a general irregular shaped region an automatic procedure is necessary to define the equivalent of the strokes of elements. To

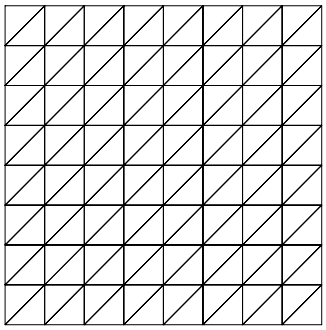

Figure 1: Rectangular domain, subdivided into triangles

that end we assume that the nodal points have been renumbered before, in order to get a small profile for example by the standard reversed Cuthill-McKee renumbering algorithm. Next we define a level structure which is very similar to the Cuthill-McKee structure. We start with node 1 and find all neighbours of this node. Let node i_1 have the highest number of this set of neighbours. Then level 1 is defined as the set of nodes 1 to i_1. The next level is found by considering all new neighbours of level 1. Let node i_2 have the highest number of this new set. Then the next level is defined as the i_1+1 to i_2. This process is repeated until all neighbours are part of a level. With respect to the start it might be necessary to combine levels 1 and 2. Per level we number first the velocities and then the pressure degrees of freedom. In this way we get a nearly optimal numbering, which may be applied in combination with Gaussian elimination. To our knowledge this type of numbering has not yet been published before. Experiments have shown that indeed this numbering is suitable for the solution of the equations by a direct method.

In the literature one often tries to solve the problems due to the *zero-block* by segregating the computation of pressure and velocity. Well known methods are for example the penalty function method, pressure correction and the method of divergence-free vector fields.

In terms of finite element methods the penalty approach is very popular. In this approach the continuity equation is disturbed by a small pressure term:

$$\varepsilon p + div\mathbf{u} = 0. \tag{5}$$

This allows us to express the pressure explicitly in the velocity and to substitute this pressure in the momentum equations. In this way an equation for the velocity only remains. A clear advantage of this approach is the reduction of the size of the system of equations

and the fact that there is no need to use a pivoting strategy. An important disadvantage of the penalty function method is that we have to choose the parameter ε in a suitable way. If ε is too large, the accuracy of velocity and pressure is insufficient, if ε is too small, the condition of the system of equations becomes too large because $1/\varepsilon$ appears in the equations. For many practical problems a choice of $\varepsilon = 10^{-6}$ is reasonable. However, even if we have a good penalty parameter still we are faced with a problem. The condition of the system of equations to be solved is very large and as a consequence it is not possible to solve the equations by an iterative method. Especially for three-dimensional problems direct methods lead to unacceptable memory requirements and computing times.

The pressure correction method is typically a method for time-dependent problems. Its application in case of finite element methods is somewhat more complex than in case of finite volumes. We will not go into details.

The method of divergence-free vector fields is quite complicated to program, especially in three dimensions. For that reason it is not commonly used.

A method that tries to overcome some of the disadvantages of the penalty function method is the so-called Uzawa scheme. In fact this is a type of iteration method that is based on the same ideas as the penalty function method. The main difference is that the parameter ε has a moderate value of order 10^{-1}. Although this method does not have the ill-conditioned property of the penalty method, in practice the convergence is not very fast. For that reason we shall investigate an iterative method that is based on the original equations.

3 Incomplete LU decompositions

In this section we consider the iterative solution of the linear system, that has been given in the previous section. We solve this system with iterative methods of Krylov subspace type combined with preconditioners based on incomplete LU decompositions [1]. First we compare two different representations of the coupled linear system. Thereafter two different ILU preconditioners are defined. Finally, some existence results are proved for the ILU decompositions.

From Section 2 it appears that when we use the p-last ordering we have to solve the following coupled system of equations:

$$\tilde{M} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}. \tag{6}$$

Another representation of this equation is:

$$M \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} A & B^T \\ -B & 0 \end{pmatrix} \begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}. \tag{7}$$

An advantage of (6) is that for the Stokes problem $\tilde{M}$ is a symmetric matrix. For Navier-Stokes A is non-symmetric, so $\tilde{M}$ is also non-symmetric. Disadvantages of (6) are: $\tilde{M}$ is never positive definite and a symmetric incomplete LL^T decomposition breaks down. First we consider the positive definiteness properties of $\tilde{M}$ and M. We have:

$$\begin{pmatrix} x^T & y^T \end{pmatrix} \begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^T A x + 2x^T B y, \tag{8}$$

and

$$\begin{pmatrix} x^T & y^T \end{pmatrix} \begin{pmatrix} A & B^T \\ -B & 0 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} = x^T A x. \tag{9}$$

For a given x the vector y can always be chosen in such a way that the right-hand side of (8) is negative. On the other hand for every choice of x and y the sign of the right-hand side of (9) only depends on the properties of A. So if A is positive definite then M is positive semi-definite.

We now consider two incomplete LU decompositions: the classic ILU decomposition [3] and ILUD, where only the diagonal is changed. Both versions are used. It appears that ILU is more robust, but more expensive, whereas ILUD is easier to analyse. The matrix M is decomposed into the following matrices: L a lower triangular matrix, D a diagonal matrix, U an upper triangular matrix. Furthermore $diag(L) = diag(U) = D$. The decomposition is made such that $\hat{L}\hat{U} \approx M$ where $\hat{L} = LD^{-1/2}$ and $\hat{U} = D^{-1/2}U$. The following rules are used:

ILU
1. $l_{i,j} = u_{j,i} = 0$ if there is no connection between i and j,
2. $(\hat{L}\hat{U})_{i,j} = m_{i,j}$ if there is a connection between i and j.

ILUD
1. $l_{i,j} = m_{i,j}$, and $u_{j,i} = m_{j,i}$ for $i > j$,
2. $(\hat{L}\hat{U})_{i,i} = m_{i,i}$.

Note that for both preconditioners $d_i > 0$ is necessary in order to form $\hat{L}$ and $\hat{U}$.

In the remaining part of this section, we give some existence results for the ILUD preconditioner. Therefore we consider the second rule from ILUD:
$(\hat{L}\hat{U})_{i,i} = (LD^{-1/2}D^{-1/2}U)_{i,i} = d_i + \sum_{j=1}^{i-1} \frac{l_{i,j} * u_{j,i}}{d_j} = m_{i,i}$.
Combination with rule 1 leads to:

$$d_i = m_{i,i} - \sum_{j=1}^{i-1} \frac{m_{i,j} * m_{j,i}}{d_j}. \tag{10}$$

Suppose $M \in I\!R^{n \times n}$ and $A \in I\!R^{n_1 \times n_1}$ then we have the following theorem:

Theorem 1 *If the ILUD decomposition of A exists and the norm of every column of B^T is non-zero, then the ILUD decomposition of M exists and $d_i > 0$ for $i \in [1, n]$.*

Proof: From the assumptions it follows that the ILUD decomposition of A exists and thus $d_j > 0$ for $j = 1, ..., n_1$. For $i \in (n_1, n]$ we have $m_{i,j} = -m_{j,i}$ and $m_{i,i} = 0$. This together with (10) implies that $d_i = \sum_{j=1}^{i-1} \frac{m_{i,j}^2}{d_j}$. Since the norm of a column of B^T is non-zero we have $\sum_{j=1}^{i-1} m_{i,j}^2 > 0$. Combined with $d_k > 0$ for $k < i$ it follows that

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

□

Remark:
The assumption on B is satisfied in many practical applications, but the assumption on A is not always satisfied. If the ILUD preconditioner is applied to $\tilde{M}$ then $d_j > 0$ for $j \leq n_1$, but $d_{n_1+1} < 0$ so it is impossible to form $\hat{L}$ and $\hat{U}$.

Suppose another ordering is used, for instance the p-last per level ordering. Then there exists a permutation matrix P such that M is given by

$$M = P^T \begin{pmatrix} A & B^T \\ -B & 0 \end{pmatrix} P.$$

The equations $m_{i,j} = -m_{j,i}$ and $m_{i,i} = 0$ again hold for a row which corresponds with a pressure unknown.

Theorem 2 *Suppose that the ILUD decomposition exists for $j < i$ (so $d_j > 0$) and the i^{th} row corresponds with a pressure unknown. If there is one $k < i$ such that $m_{i,k} \neq 0$ then $d_i > 0$.*

Proof: It follows again from (10) that

$$d_i = \sum_{j=1}^{i-1} \frac{m_{i,j}^2}{d_j}.$$

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

This implies that the ILUD decomposition does not break down ($d_i \neq 0$) in a pressure row, if every pressure unknown is preceded by a velocity unknown with a non-zero connection. Both theorems hold for Stokes and Navier-Stokes problems.

4 Numerical experiments

In this section we give some numerical experiments with the preconditioners given in Section 3. We start with the solution of the Stokes equations, where we compare the iterative method with a direct and a penalty method. Thereafter we solve the Navier-Stokes equations on a three-dimensional Backward Facing Step problem.

4.1 The Stokes equations

We consider the Stokes equations on a cube. In Table 1 we give the number of unknowns and the size of the matrices for the coupled system. Only non zero elements are stored and the rows and columns corresponding to essential boundary conditions have been removed. The ratio between these two numbers gives the average number of non-zero elements per row. It appears that this ratio is relatively large (≈ 180). This has two important implications: the CPU time for a matrix vector multiplication is large with respect to the CPU time for a vector update, and a large part of the fill-in is used in the classical ILU decomposition, so we expect a fast convergence of the preconditioned iterative method.

number of elements	number of unknowns	non-zero entries of the matrix	ratio
$3 \times 3 \times 3$	483	75 000	155
$6 \times 6 \times 6$	4 857	840 000	173
$12 \times 12 \times 12$	43 400	7 800 000	180

Table 1: The size of the problem with respect to the grid-size

In Table 2 we summarise the results for three different methods: a penalty method, a direct method and an iterative method (GMRES with ILUD). The last two methods are applied to the coupled problem (7). We observe no break down of the direct and iterative method if the p-last per level ordering technique is used (see Section 2). Comparing the different solution methods, it appears that using the iterative method leads to a large decrease in CPU time and memory requirements. The CPU time is measured in seconds on an HP 735 workstation.

Now we give a theoretical consideration of the work and memory requirements. It appears that for the penalty and direct method the memory required is proportional to n_1^5, where n_1 denotes the number of grid points in the x_1-direction. The amount of work for these methods depends on n_1^7. For the iterative method the values are n_1^3 and n_1^4 respectively. In Table 3 we compare these expressions with the measurements. For the iterative

	$3 \times 3 \times 3$		$6 \times 6 \times 6$	
method	non-zero entries	CPU time	non-zero entries	CPU time
direct	160,000	1.3	5,000,000	237
penalty	100,000	0.47	3,400,000	132
iterative	75,000	0.08	840,000	2.74

Table 2: The CPU time and the memory requirements for the various methods

method	memory	work
direct	31.7 (32)	182 (128)
penalty	34 (32)	247 (128)
iterative	11 (8)	34 (16)

Table 3: The ratio for the memory and work requirements for $n_1 = 6$ and $n_1 = 3$. Between brackets the theoretical expected values

method we were able to obtain also the ratios for $n_1 = 12$ and $n_1 = 6$. They are given by 9.4 (memory) and 24 (work). We see a reasonable correspondence between theory and experiment. Furthermore the differences between the direct and penalty method and the iterative method increases enormously for increasing grid size.

In this example both preconditioners are used. It appears that ILU is more robust than ILUD and it leads to less iterations of the preconditioned GMRES method. However the construction of the ILU decomposition takes a lot more work and doubles the memory required. For this reason if the ILUD decomposition does not breakdown ILUD is preferred, because the extra memory is negligible and the total CPU time is, in general, less than that for ILU.

4.2 The Navier-Stokes equations

In this section we solve the Navier-Stokes equations on a three-dimensional Backward Facing Step problem. The geometry is given in Figure 2. At the left boundary surface we use a Dirichlet inflow boundary condition and at the right boundary surface we use an outflow boundary condition: $\sigma_{nn} = 0$ and $\sigma_{\mathbf{nt}} = \mathbf{0}$. At all other boundaries we use a no slip condition. In this case we have to solve a non linear problem. The strategy to do this is given in Section 2. Initially we solve the corresponding Stokes equations, thereafter some Picard iterations are done and finally some Newton Raphson iterations are used.

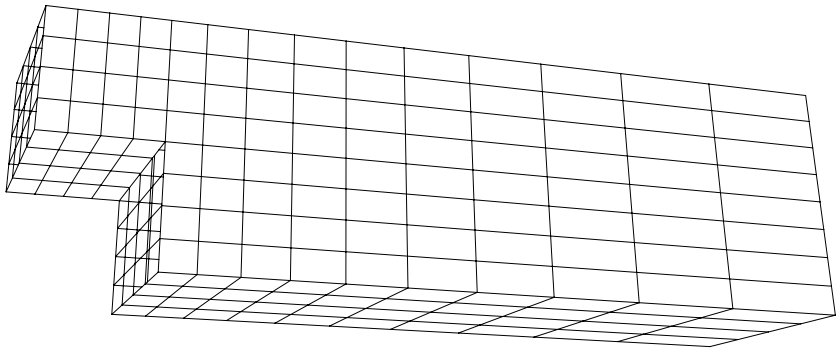

Figure 2: The geometry of the three-dimensional Backward Facing Step problem

The ILU/ILUD decompositions are based on the current coefficient matrix, so a new decomposition is made in every outer iteration. As Krylov subspace methods we use GMRES [4] and Bi-CGSTAB [5]. When we restrict ourselves to Stokes and Picard outer iterations then for both methods 9 outer iterations are needed. The total number of inner iterations is 153 for GMRES and 122 for Bi-CGSTAB. The total CPU time (including building of the matrices and decompositions) is 351 s for GMRES and 385 s for Bi-CGSTAB. Note that Bi-CGSTAB uses less iterations, but one iteration of Bi-CGSTAB is approximately two times as expensive as an iteration of GMRES. This explains the bigger CPU time for Bi-CGSTAB. The optimal CPU time for GMRES is not unexpected since it is known that if the number of iterations is small and a matrix vector product is expensive (which means a large number of non zero elements per row) then GMRES is the best method (see [6]).

We have experimented with different strategies to solve the non linear equations. It appears that the number of inner iterations in a Newton Raphson step is slightly more than in a Picard step. However in general less outer iterations are needed if Newton Raphson is used. For this reason we use the Stokes equations in the first iteration, Picard in the second iteration and Newton Raphson in the following iterations. In general 5 or 6 outer iterations are sufficient to reduce the initial error with a factor of 10^{-4}. We stop the inner iteration if $\|r_k\|_2/\|r_0\|_2 < eps$. If Picard iterations are used $eps = 10^{-1}$ is sufficient. If Newton Raphson steps are used it may be better to use $eps = 10^{-2}$, because then the outer iterations converge quadratically, whereas if $eps = 10^{-1}$ is used Newton Raphson has a linear convergence behaviour.

Finally we observe that in this problem the ILUD decomposition breaks down, so we only

use the ILU preconditioner. It appears that break down of the ILUD decomposition always happens in the velocity part, which is in agreement with the theory given in Section 3. The construction of the ILU decomposition is expensive. The CPU-time is comparable with the CPU time to build the coefficient matrix. With respect to the ordering techniques we observe no break down of the ILU decomposition using the p-last, or p-last per level ordering. It appears that the p-last per level ordering leads to less inner iterations and CPU time than the p-last ordering.

We end this section with some results for the BFS problem on an $8 \times 16 \times 28$ grid. The number of unknowns is equal to 8×10^4 and the number of non zero entries of the matrix is equal to 1.4×10^7. The CPU time to build the matrix is 2 min., whereas the CPU time to build the ILU preconditioner is 3 min. Using ILU, GMRES, p-last per level and $eps = 10^{-2}$, 5 outer iterations are needed. The total CPU time is 80 min. and the total number of inner iterations is 350.

5 Conclusions

In this paper the incompressible Navier-Stokes equations are solved. We consider the momentum equations coupled with the continuity equation.

Ordering techniques are described to prevent break down of the LU decomposition. From our numerical experiments it appears that direct methods can be used with the p-last per level ordering. The CPU time and memory requirements for the direct method are comparable to that of the penalty approach.

Thereafter the proposed orderings are combined with preconditioned Krylov subspace methods. It appears from the theory that the zero main diagonal elements no longer lead to break down or bad convergence. Furthermore, it appears that the iterative methods are much better, than direct or penalty methods.

ACKNOWLEDGEMENT

The authors would like to thank C.G.M. Kassels for software support.

References

[1] R. Barrett, M. Berry, T.F. Chan, J. Demmel, J. Donato, J. Dongarra, V. Eijkhout, R. Pozo, C. Romine, and H. van der Vorst. *Templates for the solution of linear systems.* SIAM, Philadelphia, 1994.

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

[3] J.A. Meijerink and H.A. Van der Vorst. An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. *Math. Comp.*, 31:148–162, 1977.

[4] Y. Saad and M.H. Schultz. GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems. *SIAM J. Sci. Stat. Comp.*, 7:856–869, 1986.

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

[6] C. Vuik. Further experiences with GMRESR. *Supercomputer*, 55:13–27, 1993.