Experiments with MRILU for linear systems in PARNASSOS

Elke Ottenhoff

Department of Mathematics

RuG

Master's Thesis

Experiments with MRILU for linear systems in PARNASSOS

Elke Ottenhoff

Supervisor:
Dr.ir. F.W. Wubs
Department of Mathematics
University of Groningen
P.O. Box 800
9700 AV Groningen

August 2004

Contents

Introduction

MARIN, the Maritime Research Institute Netherlands is a research institute in hydrodynamics and nautical research. The MARIN has developed two codes, called RAPID and PARNASSOS, for predicting the flow around the hull. RAPID predicts the steady inviscid flow around a ship hull, the wave pattern and the wave resistance. PARNASSOS predicts the flow over a ship hull; it is concerned with the solvation of the (Reynolds-averaged) Navier-Stokes (RANS) equations. It predicts the velocity field, streamline patterns, and pressure distribution, and is used e.g. to eliminate separation phenomena and predict the wake field.

Discretization of the Navier-Stokes equations and a linearization process of the obtained system leads to a large sparse system of linear equations $\mathbf{Ax} = \mathbf{b}$, for a non-singular $N \times N$ matrix $\boldsymbol{A}$ and a given vector $\mathbf{b}$. A part of the code PARNASSOS is occupied with the solvation of this system. To increase the robustness and efficiency of PARNASSOS, the solution algorithms have been thoroughly revised and improved. In order to improve the applicability of PARNASSOS, continuously investigations in numerical methods are required. This thesis is concerned with an effective method to solve this system.
First a rough description of the current solution strategy will be given, which is a coupled incomplete LU-decomposition which is used as a preconditioner for GMRES. Hereafter, the matrix renumbering ILU (MRILU) factorization, will be described, together with a discussion whether or not it is suitable for implementation in PARNASSOS.
This thesis is build up as follows. The first chapter will give a short revision of the
Navier-Stokes equations, which are the fundamental partial differential equations that describe the flow of incompressible fluids (in this case water). An illustration of the computational grid, used to solve the (time-averaged) Navier-Stokes equations, will be given and the way the accompanying coefficient matrix $\boldsymbol{A}$ is contructed will be explained.
In chapter 2, the currently used method to solve the linear system $\mathbf{Ax} = \mathbf{b}$, the advantages of preconditioning and the MRILU method will be explained.
Chapter 3 will deal with test cases, that have been carried out with both preconditioned GMRES and MRILU; based on the gained results, a comparison between these two methods has been be made.
Finally, some conclusions will be given and some suggestions for future research will be made. An outline of the theory used in this thesis is given in the appendix.

Chapter 1

Computational Model

1.1 Navier-Stokes equations

The Navier-Stokes equations describe how the velocity, pressure, temperature and density of a moving fluid are related. These equations consists of a time-dependent *continuity equation* for *conservation of mass* and three (for each of the three coordinates) time-dependent *conservation of momentum* equations.
In this case the moving fluid is water. Water can be considered as an incompressible and viscous fluid. For the water motion, the Navier-Stokes equations describing the conservation of mass and momentum can be simplified to

- *conservation of momentum*

$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho}\nabla p + \nu \nabla^2 \boldsymbol{u} \tag{1.1}$$

- *conservation of mass (continuity equation)*

$$\nabla \cdot \boldsymbol{u} = 0 \tag{1.2}$$

where $\boldsymbol{u}$ is the velocity vector, p the pressure, ρ the density and ν the kinematic viscosity.

MARIN has developed his own code, called PARNASSOS, to solve the (steady) Navier-Stokes, or actually, for practical reasons, the time-averaged form of this model. The pertinent equations of motion are referred to as the Reynolds-averaged Navier-Stokes (RANS) equations. Where needed, these equations are suplemented with a turbulence model.

1.2 Computational Grid

PARNASSOS solves the steady Reynolds-averaged Navier-Stokes equations, in a structured, body-fitted curvilinear grid, using a finite-difference method. Viscous effects are noticeable

only in a thin layer around the hull and in the wake. Further away from the hull, the fluid behaves as being essentially inviscid, which allows a considerable reduction of the complexity of the mathematical model. On the forward part of the ship the viscous layer is so thin that the boundary layer equations are an adequate description. The solution of the RANS equations can therefore be restricted to a relatively small part of the fluid domain, enclosing the aft half of the ship and a part of its wake.
If the geometry of the ship is not too complicated, the computational grid at the afterbody of the ship can be represented by the grid in Fig. 1.2.2. In here, the location of the ship hull is shown. Imagine now that the ship moves to the right, while the (main-stream) flow of the water is in the opposite direction. This direction is denoted by the Greek letter $\boldsymbol{\xi}$. Also a wall-normal direction $\boldsymbol{\eta}$ and girthwise direction $\boldsymbol{\zeta}$ are depicted. These directions form the three dimensional curvilinear coordinate system ($(\boldsymbol{\xi},\boldsymbol{\eta},\boldsymbol{\zeta})$-system). The number of grid nodes are denoted by NX, NY and NZ, which are in main-stream, wall-normal and in girthwise direction, respectively.

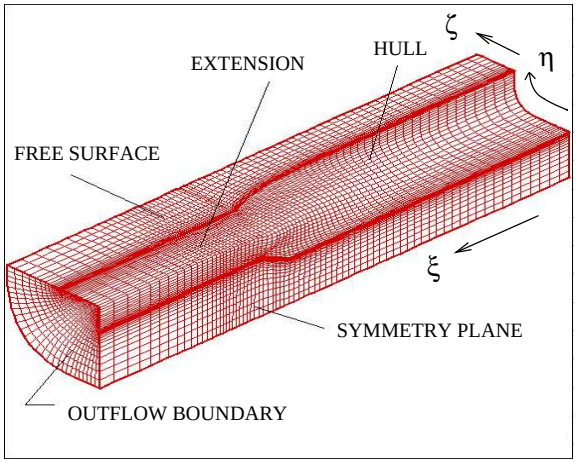

Figure 1.1: *Computational Grid*

A detailed description of the mathematical model, the computational grid and de PDEs in curvilinear coordinates is given in the master thesis of Dr Ir Hoekstra [2].

1.2.1 Sub-domains

Discretization of the equations leads to an enormous system of non-linear algebraic equations, to which a linearization process is applied. In order to reduce the size of equations,
PARNASSOS makes use of a so-called *marching solution scheme*. For this, the computational domain is divided into *sub-domains*. Each sub-domain is a plane that is nearly orthogonal to the main-stream flow, a so-called $\boldsymbol{\xi}$-constant plane. The grid planes are visited in down stream order. The velocity and pressure in each grid point of such a $\boldsymbol{\xi}$-constant plane are solved simultaneaously. One *sweep* through the domains, is called *global iteration*.

An advantage of this plane-by-plane marching approach is that it yields enormous savings in memory requirements. However, on the other hand, the upstream communication of down-

stream occurences may proceed rather slowly via the global iteration process. A way to speed up the solvation of the linear system of equations is to use a proper preconditioning technique. Such a preconditioning technique will be explained in section 2.2.2.
Another option is to choose for larger subdomains. This is described in the next section.

1.2.2 Larger Sub-domains

The single-grid-plane sub-domain has originally been chosen to minimize memory requirements, but on present-day machines, memory is amply available. Therefore, PARNASSOS is able to reduce the time that is needed for this process to make use of larger sub-domains. Instead of solving the equations for only one $\boldsymbol{\xi}$-constant plane, a sub-domain is used in which more planes are taken into account. Now, the variables in one sub-domain are solved simultaneously.

The size of a sub-domain is defined by a number (say g, $g \geq 1$) of consecutive grid-planes. Two of these sub-domains are depicted with lines in the figure below.

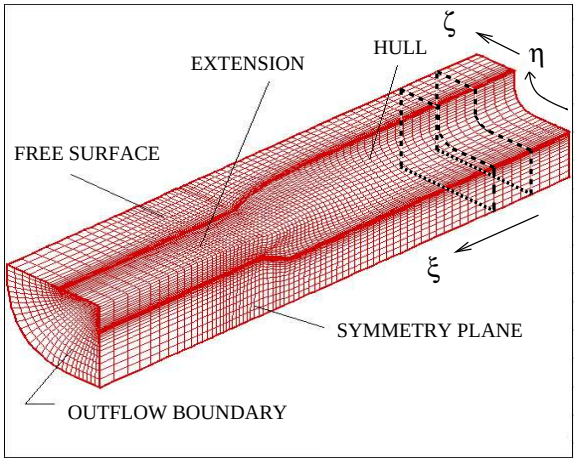

Figure 1.2: *Computational Grid: larger sub-domains*

While in the plane-by-plane solution strategy several steps of the linearization process in a sub-domain are needed to let the global iteration safely converge, it turned out that with $g > 1$ it's possible to make only one step of the linearization process before going to the next sub-domain. Hence the global iteration accounts completely for the non-linearity, while a significant saving in computation time is obtained [1].

Next to the global iteration, in which the sub-domains are visited in downstream direction, an inner iteration process is needed in each sub-domain for the linearization process. During this linearization process a *Jacobian matrix*[4] is constructed, which is denoted by matrix $\boldsymbol{A}$. Matrix $\boldsymbol{A}$ contains terms representing the coupling between the g planes covered by the sub-domain. If a larger sub-domain is used, matrix $\boldsymbol{A}$ is g times bigger than in the plane-by-plane approach of PARNASSOS. This coefficient matrix has a special structure, which is explained in the next section.

1.3 The Coefficient Matrix $\boldsymbol{A}$

As a result of the chosen discretization [2], the coefficient matrix $\boldsymbol{A}$ has the following penta-diagonal structure.

$$\begin{bmatrix} d_1 & u_1 & v_1 & 0 & \cdots & 0 \\ s_2 & d_2 & u_2 & v_2 & \cdots & 0 \\ t_3 & s_3 & d_3 & u_3 & \ddots & 0 \\ 0 & t_4 & s_4 & d_4 & \ddots & \ddots \\ \vdots & \vdots & \ddots & \ddots & \ddots & \ddots \\ 0 & 0 & 0 & t_g & s_g & d_g \end{bmatrix} \tag{1.3}$$

The oefficients of matrix $\boldsymbol{A}$ account for the coupling between the seperate $\boldsymbol{\xi}$-constant planes in a sub-domain. As mentioned before, if the size of a sub-domain is g, the coefficient matrix $\boldsymbol{A}$ is g times larger than in the plane-by-plane version of PARNASSOS. The coefficients of matrix $\boldsymbol{A}$ can be represented by matrices again, in which all elements multiply the variables in a $\boldsymbol{\xi}$-constant plane.
The entries of the latter have a 4×4 block structure. in which the first row represents the momentum equation (eq. (1.1)) in main-stream direction. The second represents the continuity equation (eq. (1.2)) and the third and fourth row represents the momentum equation in girthwise direction and wall-normal direction, respectively.

Usually, the coefficients $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ in matrix (1.3) only contain elements coming from the discretization of the pressure derivative in main-stream direction. Only in the case of flow separation[2], these matrix elements can also contain small contributions from the derivative of the velocities in main-stream direction.
The effect of these coefficients on the solvation of the linear system of equations will be described in chapter 3.
Now, with a given matrix $\boldsymbol{A}$ and a right-handside vector $\boldsymbol{b}$, it's time to solve the linear system $\boldsymbol{Ax} = \boldsymbol{b}$. This is described in the next chapter.

Chapter 2

Solving the Linear Systems $Ax = b$

2.1 The Linear System $Ax = b$

In this chapter, the large sparse[9] system of linear equations[6] of the form

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{2.1}$$

will be considered. In here, matrix $\boldsymbol{A}$ is a real sparse non-singular $N \times N$ matrix, which, as mentioned in the previous chapter, is the Jacobian matrix that contains coefficients which account for the coupling between the seperated $\boldsymbol{\xi}$-constant planes.
The aim is to obtain a solution $\boldsymbol{x}$ for this matrix $\boldsymbol{A}$ and a given vector $\boldsymbol{b}$ on a suitable way.
One way to solve (2.1) is to use a *direct* method, for example, by the constuction of an LU-decomposition[??] (or LU-factorization). In this approach an upper-triangular part $\boldsymbol{U}$ and a lower-triangular part $\boldsymbol{L}$ are constructed in such a way that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$. The system (2.1) can be solved in two steps using the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ subsequently. This means to solving

$$\boldsymbol{A}\boldsymbol{x} = (\boldsymbol{L}\boldsymbol{U})\boldsymbol{x} = \boldsymbol{L}(\boldsymbol{U}\boldsymbol{x}) = \boldsymbol{b} \tag{2.2}$$

by first solving $\boldsymbol{L}\boldsymbol{y} = \boldsymbol{b}$ for $\boldsymbol{y}$ and then solving $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{y}$ for $\boldsymbol{x}$.
One major drawback of this strategy is the fact that $\boldsymbol{L}$ and $\boldsymbol{U}$ are not sparse due to fill-in during the factorization process, so that computer storage demands are very high. Moreover, the computational work for constructing the factor $\boldsymbol{L}$ and $\boldsymbol{U}$ increases strongly with the dimension of the problem.

As a result, systems like (2.1) are often solved with *iterative* methods, which generates a sequence of approximations $\boldsymbol{x}^{(n)}$ which hopefully converge rapidly towards the exact solution $\boldsymbol{x}$. Iterative techniques have a number of advantages over direct solution: first of all, the often can exploit the sparsity of the coefficient matrix and secondly, the accuracy of the solution can be controlled more easily [3].

In the current version of PARNASSOS, the (iterative) solution strategy, is to construct a coupled incomplete LU-decomposition which is used as a preconditioner for GMRES. The terms GMRES and preconditioning will be described in the next sections.
Hereafter, another kind of preconditioner, the matrix renumbering ILU (MRILU) factorization will be considered.

2.2 The Original Solution Strategy

2.2.1 Generalized Minimal Residual Method

An iterative method, suited for non-symmetric matrices, is the *generalized minimal residual (GMRES) method*. The GMRES method solves the system projected onto the Krylov subspace[5] $K_k(\boldsymbol{A}; \boldsymbol{r}^{(0)})$. GMRES is used to minimize the number of matrix-vector multiplications, which is a very expensive operation in iterative methods. At the k-th iteration step of GMRES, $\boldsymbol{x}^{(k)}$ is the vector that minimizes $||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{y}||_2$ over all $\boldsymbol{y} \in \boldsymbol{x}^{(0)} + K_k(\boldsymbol{A}; \boldsymbol{r}^{(0)})$.
This has the advantage that the 2-norm[8] of the residual vector does not increase when the iteration process proceeds, and with exact arithmetic the method terminates within N steps. At every step of GMRES an orthonormal basis $\{\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \cdots, \boldsymbol{v}^{(k)}\}$ for $K_k(\boldsymbol{A}; \boldsymbol{r}^{(0)})$ has to be formed. If $\boldsymbol{V}^{(k)}$ denotes the matrix with columns $\{\boldsymbol{v}^{(1)}, \boldsymbol{v}^{(2)}, \cdots, \boldsymbol{v}^{(k-1)}\}$, this leads to

$$\boldsymbol{A}\boldsymbol{V}^{(k)} = \boldsymbol{V}^{(k+1)}\boldsymbol{H}^{(k)} \tag{2.3}$$

where $\boldsymbol{H}^{(k)}$ is an $(k+1) \times k$ upper-Hessenberg matrix[3].
A disadvantage of GMRES is that all the vectors $\boldsymbol{v}^{(k)}$ have to be stored in memory, and the construction of an orthonormal basis for the Krylov subspace becomes more complex with increasing k. Therefore, GMRES is restarted after M steps, so that one obtains an *iterative* method GMRES(M) instead of a *direct* method [3], [7], [9].

2.2.2 Preconditioning

The speed of convergence of GMRES strongly depends on the eigenvalue spectrum[10] of $\boldsymbol{A}$. Ideally, the matrix should have all eigenvalues clustered; then $\boldsymbol{A}$ would be close to the identity matrix and GMRES will rapidly converge. In order to bring a given spectrum close to that ideal situation, one can use a *preconditioning* technique [1].
Preconditioners are approximate solvers to some problems. Instead of solving $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$, one solves $\boldsymbol{M}^{-1}\boldsymbol{A}\boldsymbol{x} = \boldsymbol{M}^{-1}\boldsymbol{b}$, in which the non-singular matrix $\boldsymbol{M}$ is called the *preconditioner*. The preconditioner should have the following properties [3].

1. It should be a proper approximation of $\boldsymbol{A}$, so that $\boldsymbol{M}^{-1}\boldsymbol{A}$ resembles the identity matrix. The most important quality of a preconditioner is to reduce the spectral condition number of the preconditioned matrix[11] $\boldsymbol{M}^{-1}\boldsymbol{A}$.

2. The preconditioner should be cheap to compute, and it should be possible to solve the system $\boldsymbol{M}\boldsymbol{y} = \boldsymbol{d}$ for given $\boldsymbol{d}$ in $O(N)$ operations.

3. It should not require a large amount of storage.

There are several possibilities for choosing a preconditioner. The quality of the preconditioner determines the success of the iterative method.

More than 80 % of the CPU-time is spent on the solution of the system of linear equations, using preconditioned GMRES [1]. Results of a number of tests have shown that solving the linear system with the present implemented method is reasonable good.
Still, improvements could be made. For example, with the current implementation, treating 4 planes in a sub-domain simultaneously, seems to be a good choice in many practical situations. The speed of convergence is not improved by choosing the value of g larger than 4. However, to increase the size of sub-domains is very likely, since it reduces the number of global iteration steps.
This is one of the reasons for trying another method.

An effective algebraic multilevel ILU preconditioner for sparse matrices [4], the so-called matrix renumbering ILU (MRILU) method has been investigated. It is a preconditioner that is based on an ordering of the unknowns and a dropping strategy employed during the factorization. Due to its positive results for a number of problems and its attractive properties, it seems a good alternative for the present implemented method. In the next section, a description of the MRILU method will be given, together with its properties.

Chapter 3 will deal with test cases that have been carried out with the current method and with MRILU. The gained results of these test cases should give an indication whether or not the MRILU method is suitable to be integrated into PARNASSOS.

2.3 Matrix Renumbering ILU

Many preconditioning techniques are based on an *Incomplete* LU-factorization of the coefficient matrix, because of its simple definition and high efficiency [7]. In an ILU-factorization, the preconditioning matrix $\boldsymbol{M}$ (§2.2.2) is chosen equal to $\boldsymbol{LU}$, such that

$$\boldsymbol{A} = \boldsymbol{LU} + \boldsymbol{R}$$

which can be treated as a standard LU-decomposition, but in which the sparsity of the factors $\boldsymbol{L}$ and $\boldsymbol{U}$ is preserved by ignoring some or all elements causing fill-in additional to that of $\boldsymbol{A}$. These elements are moved to the residual matrix $\boldsymbol{R}$. The residual matrix $\boldsymbol{R} = \boldsymbol{A} - \boldsymbol{LU}$ is useful as an estimate for the quality of the preconditioner.

It has been observed that in any (I)LU factorization an *ordering* may have a significant impact on the amount of fill. Finding an optimal ordering is difficult, but over the years several successful heuristics have evolved. It turns out that ordering can have dramatic influence on the convergence [4].

The so-called matrix renumbering ILU (MRILU) method determines the ordering by itself. The *ordering* is carried out during the factorization process. Together with *dropping*, which means that fill is dropped when its value is smaller than some threshold, they form the essential ingredients of the MRILU method. The construction of the MRILU method is described in section 2.3.1. Thereafter, a few important characteristics of the MRILU method will be given.

2.3.1 Construction of MRILU factorization

Below a description of the ordering and dropping strategy of the MRILU preconditioner.
The ordering is determined during the construction of the factorization and is based on the sparsity pattern of the matrix and the magnitude of the elements. During the factorization small elements are dropped resulting in an incomplete LU-factorization. To decide whether or not an element will be dropped, a *lump space* ϵ is determined which is based on the diagonal of the factorization. To make sure the lump space is not consumed too fast some additional restrictions are made.
In Algorithm 1 the steps of the construction of the MRILU factorization are given. Hereafter, this algorithm is explained [11].

Algorithm 1 Construction of MRILU factorization

$A^{(0)} = A$

for $i = 1, \cdots, M$

1. Make a reordering and partitioning of $A^{(i-1)}$

$$\begin{pmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} \end{pmatrix}$$

2. Approximate $\boldsymbol{A}_{11}$ by a diagonal matrix $\tilde{A}_{11}$

3. Drop small elements of $\boldsymbol{A}_{12}$ and $\boldsymbol{A}_{21}$

4. Make an imcomplete factorization

$$\begin{pmatrix} I & 0 \\ \tilde{A}_{21}\tilde{A}_{11}^{-1} & I \end{pmatrix} \begin{pmatrix} \tilde{A}_{11} & \tilde{A}_{12} \\ 0 & A^{(i)} \end{pmatrix}$$

where $A^{(i)} = \boldsymbol{A}_{22} - \tilde{A}_{21}\tilde{A}_{11}^{-1}\tilde{A}_{12}$ (=Schur complement)

end

Make an exact or incomplete factorization of $A^{(M)}$

In step 1 a reordering of the unknowns and a partitioning of the so-called *Schur complement* $A^{(i-1)}$ is made. Assume the factorization has progressed to the point that the Schur complement $A^{(i-1)}$ has been computed. The steps needed to compute the next Schur complement will be explained.
For sparse matrices the unknowns can be divided such that the matrix $\boldsymbol{A}_{11}$ is diagonal, i.e. the unknowns of $\boldsymbol{A}_{11}$ form an independent set. By allowing also weak connections between the unknowns of $\boldsymbol{A}_{11}$, i.e. they form a nearly independent set, $\boldsymbol{A}_{11}$ can be enlarged. The partitioning is made in such a way that the matrix $\boldsymbol{A}_{11}$ is strictly *diagonally dominant*[1], i.e. the elements of $\boldsymbol{A}_{11}$ satisfy the following definition.

Definition 2.1 (Diagonally Dominant) *A real $N \times N$ matrix $\boldsymbol{A}$ is said to be diagonally dominant if*

$$a_{ii} \geq \sum_{j=1_{j \neq i}}^{N} |a_{ij}| \quad for\ all\ i = 1, \cdots, N \tag{2.4}$$

The matrix is said to be strictly diagonally dominant, wahen '$\geq$' is replaced by '$>$'.

The ordering of the unknowns is constructed by a greedy algorithm. By keeping track of the absolute sum of the columns belonging to the unknowns selected for $\boldsymbol{A}_{11}$, it can easily be decided whether or not to add a new unknown to the near independent set selected so far.

In step 2 the matrix $\boldsymbol{A}_{11}$ is replaced by a diagonal matrix $\tilde{A}_{11}$. This is an accurate approximation because by construction the matrix $\boldsymbol{A}_{11}$ is strongly diagonally dominant. The approximation simplifies the construction of the next Schur complement $A^{(i)}$. The unknowns belonging to $\boldsymbol{A}_{11}$ can be eliminated simultaneously, which means that the ordering of these unknowns is not important.

In step 3 small elements will be dropped to limit the number of nonzero elements in the factorization. The dropping outside $\boldsymbol{A}_{11}$ is limited to $\boldsymbol{A}_{12}$ and $\boldsymbol{A}_{21}$. Gustafsson's modifcation can be used when an element is dropped. Gustafssons's modification means that the dropped element is added to the diagonal. When an element a_{ij} is dropped row i and column j are modified. Whether or not it is acceptable to drop this element is measured by the ratio of the element a_{ij} and the diagonal elements a_{ii} and a_{jj} and on the amount dropped so far in row i and column j.
To obtain an accurate dropping strategy these criteria have to be adjusted. A better strategy is obtained when the dropping criteria are not based on the diagonal of $\boldsymbol{A}_{22}$ only. The diagonal of the next Schur complement $A^{(i)}$ may differ signifcantly from the diagonal of $\boldsymbol{A}_{22}$, especially for matrices with strongly varying elements. Relatively small modifications compared to the diagonal of $\boldsymbol{A}_{22}$ may be large compared to this new diagonal. Therefore, the dropping criteria should be based on the diagonal D of the complete factorization. The diagonal of $\boldsymbol{A}_{11}$ becomes (slightly modified) a part of D. Therefore, dropping elements outside $\boldsymbol{A}_{11}$ is much more critical. The diagonal D of the complete factorization is not known. Within $\boldsymbol{A}_{22}$ it is approximated by the diagonal of the next Schur complement, which is temporarily calculated without the dropping outside $\boldsymbol{A}_{11}$. An element on row or column i is dropped when the sum of the absolute values of all discarded elements on this row or column (including those discarded on earlier levels) is smaller than $\epsilon|\boldsymbol{d}_i|$. With $\boldsymbol{d}_i$ the 'updated' diagonal element on row i. The available space for dropping has to be restricted even further. When dropping an element a_{ij} of $\boldsymbol{A}_{21}$ row i is modified. On subsequent levels of the construction more elements of this row may be discarded. Therefore, the row space for dropping within $\boldsymbol{A}_{21}$ is restricted by multiplying the remaining space by the number of columns of $\boldsymbol{A}_{21}$ divided by the dimension of $\boldsymbol{A}$. Furthermore, the available space for lumping on rows of $\boldsymbol{A}_{21}$ should not be consumed by a few large elements but rather by many small elements. Therefore, only entries smaller than a certain fraction of this space are dropped. Similar restrictions are made for the dropping within columns of $\boldsymbol{A}_{12}$.
Finally, in step 4 the incomplete factorization can be computed, resulting in the next Schur complement. Step 1 to 4 can be repeated until the final Schur complement is of low order. Then, the final Schur complement can be factorized by a standard (I)LU factorization [11].

Matrices arising from the discretization of a system of partial differential equations can be written in block form, the elements of the matrix are small blocks instead of scalars. For these matrices a block form of the MRILU factorization can be used. To decide whether or not an entry of a block row can be dropped, this block row is multiplied from the left with the inverse of the corresponding diagonal block. This can be somewhat simplified by considering only the absolute maximum in each column of this inverse. The dropping in a column can be handled in a similar way using multiplication from the right [11].
In the current situation, the concerning blocks are of size 4. This is due to the structure of the entries of the coefficient matrix $\boldsymbol{A}$; the elements are grouped in square blocks of size 4, which has been described in §1.3.

Below, a few characteristics of the MRILU method.

- MRILU works well on both structured as unstructured grids, since both ordering and dropping are based on the size of the entries of the matrix and not on the grid;
- The MRILU method has been applied successfully to symmetric, unsymmetric and indefinite problems;
- An attractive property of the MRILU method is that its structure is simple: it is merely an ILU factorization;
- For a number of problems (f.e. for the Poisson problem [11]) convergence behavior is observed that is nearly independent of the mesh size, an attractive property for very large problems.

A more detailed description of the MRILU method can be found in [4].
The MRILU factorization has succesfully been applied in a variety of problems. It had been tested on systems occuring from different types of the Navier-Stokes equations [11]. Due to the succes of the MRILU method, it has been tested for the present situation. In the current implementation of the MRILU code, the MRILU preconditioner is combined with GMRES as linear solver. The achieved test results with the MRILU preconditioner are described in the next section.

However, next to the method described above, in which MRILU is applied to the entire system of linear equations (2.1), MRILU has been applied in another way also.
The latter is a result of the special kind of preconditioning technique that is used in the current solution strategy. In this preconditioning technique, which is described in [1], the upper triangular blocks of matrix $\boldsymbol{A}$ (Eq. (1.3)) $\boldsymbol{u}_i$ and $\boldsymbol{v}_i$ are neglected during the construction of the preconditioner. As mentioned in §1.3, these terms appear due to the discretization of the pressure derivative in main-stream direction, which can contain only in the case of flow seperation small contributions from the derivative of the velocities in main-stream direction. Since these terms are small, they are neglected in the preconditioner matrix, i.e. one exploits the fact that the equations are almost parabolic in the streamwise direction.
From this concept, the idea has been arised to split up the coefficient matrix $\boldsymbol{A}$ in g blocks first and than proceed with a *Gauss-Seidel* iteration process over de g block systems, in which the small terms on the upper diagonals has been moved to the right-hand side of the system. During this Gauss-Seidel loop, MRILU is applied g times; to each of the g block (corresponding to a grid plane) systems.
In the next section, the Gauss-Seidel method will be described, together with a more detailed description of the above.

2.3.2 MRILU combined with a Gauss-Seidel Iteration Process

The Gauss-Seidel method is as follows.

(The Gauss-Seidel Method) *The Gauss-Seidel method is a well-known (stationary) iterative technique for solving the N equations of the linear system of equations* $\boldsymbol{Ax} = \boldsymbol{b}$ *one at a time in sequence, and uses previously computed results as soon as they are available,*

$$x_i^{(k)} = \frac{b_i - \sum_{j<i} a_{ij} x_j^{(k)} - \sum_{j>i} a_{ij} x_j^{(k-1)}}{a_{ii}}. \tag{2.5}$$

In terms of matrices, the definition of the Gauss-Seidel method can be expressed as

$$\boldsymbol{x}^{(k)} = (\boldsymbol{D} - \boldsymbol{L})^{-1}(\boldsymbol{U}\boldsymbol{x}^{(k-1)} + \boldsymbol{b}), \tag{2.6}$$

where the matrices $\boldsymbol{D}$, $-\boldsymbol{L}$ *and* $-\boldsymbol{U}$ *represent the diagonal, strictly lower triangular, and strictly upper triangular parts of* $\boldsymbol{A}$*, respectively.*
The Gauss-Seidel method is applicable to strictly diagonally dominant (definition (2.1)) matrices $\boldsymbol{A}$*.*

In the present situation, the $\boldsymbol{x}^{(k)}$ represents a part of the complete solution vector. This means that if $\boldsymbol{A}$ can be devided into g blocks, there are g vectors $\boldsymbol{x}_i^{(k)}$ (where $i = 1, \cdots, g$) which form the complete solution vector $\boldsymbol{x}^{(k)}$. The parts of the solution vector are corrected in the sequence $i = 1, \cdots, g$; this means that the $\boldsymbol{x}_1^{(k+1)}, \cdots, \boldsymbol{x}_{i-1}^{(k+1)}$ are already available when $\boldsymbol{x}_i^{(k+1)}$ is determined. To determine a part of the solution vector at time $k+1$, thus $\boldsymbol{x}_i^{(k+1)}$, the GS iteration uses parts of the solution vector at time k, thus $\boldsymbol{x}_{i+1}^{(k)}, \cdots, \boldsymbol{x}_g^{(k)}$ as well as parts of the solution vector at time $k+1$, thus $\boldsymbol{x}_1^{(k+1)}, \cdots, \boldsymbol{x}_{i-1}^{(k+1)}$. Since GS also uses the revised solutions, it makes it a fast iterative method.
The algorithm for determining the solution vector $\boldsymbol{x}$ by this block approach is given below.

First, suppose there is a $N \times N$ matrix $\boldsymbol{A}$, which contains g square blocks, say $\boldsymbol{A}_{ii}$, of size N

$$\boldsymbol{A} = \begin{pmatrix} \boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{1g} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{21} & \cdots & \boldsymbol{A}_{gg} \end{pmatrix} \tag{2.7}$$

and a given vector $\boldsymbol{b}$, such that the system

$$\begin{pmatrix} \boldsymbol{A}_{11} & \cdots & \boldsymbol{A}_{1g} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{21} & \cdots & \boldsymbol{A}_{gg} \end{pmatrix} \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_g \end{pmatrix} = \begin{pmatrix} \boldsymbol{b}_1 \\ \vdots \\ \boldsymbol{b}_g \end{pmatrix} \tag{2.8}$$

have to be solved.

Then, the Gauss-Seidel (GS) loop over the g blocks is as in Algorithm 2.

Algorithm 2 Construction of GS loop over g blocks

Input: $\boldsymbol{A}, \boldsymbol{b}$
$\boldsymbol{x}^{(0)} = \boldsymbol{x}_0$
$norm = ||\boldsymbol{r}^{(n)}||_2 = ||\boldsymbol{b} - A\boldsymbol{x}^{(n)}||_2$

for $i = 2, \cdots, g-1$

1. Split up matrix $\boldsymbol{A}$ into three parts.

An upper part: $U_i = (\boldsymbol{A}_{i,i+1}, \cdots, \boldsymbol{A}_{i,g})$
A lower part: $L_i = (\boldsymbol{A}_{i,1}, \cdots, \boldsymbol{A}_{i,i-1})$
A diagonal part: $D_i = \boldsymbol{A}_{i,i}$

with
$\boldsymbol{U}_1 = (\boldsymbol{A}_{1,2}, \cdots, \boldsymbol{A}_{1,g}), \quad \boldsymbol{L}_g = (\boldsymbol{A}_{i,1}, \cdots, \boldsymbol{A}_{i,i-1}),$
$\boldsymbol{D}_1 = \boldsymbol{A}_{1,1}, \quad \boldsymbol{D}_g = \boldsymbol{A}_{g,g}$

2. Write

$$D_i \boldsymbol{x}_i^{(n+1)} = \boldsymbol{b} - U_i \boldsymbol{x}_i^{(n)} - L_i \boldsymbol{x}_i^{(n)}$$

3. Apply MRILU to

$$D_i \boldsymbol{x}_i^{(n+1)} = \boldsymbol{v}_i,$$

where

$$\boldsymbol{v}_i = \boldsymbol{b} - U_i \boldsymbol{x}_i^{(n)} - L_i \boldsymbol{x}_i^{(n)}$$

4. Obtain the solutionvector $\boldsymbol{x}$

$$\boldsymbol{x}^{(n+1)} = (\boldsymbol{x}_1^{(n+1)}, \cdots, \boldsymbol{x}_g^{(n+1)})$$

end

Obtain $norm = ||\boldsymbol{r}^{(n+1)}|| = ||\boldsymbol{b} - A\boldsymbol{x}^{(n+1)}||_2$.
If $norm$ smaller than ϵ, leave GS loop; solution vector $\boldsymbol{x}$ has been determined.

Since the convergence of the Gauss-Seidel iteration may still be slow for large systems of equations [7], it's desirable to apply MRILU to the complete linear system of equation directly. However, a problem occurs with this solution strategy. This might become clear in the next chapter, in which the tests that have been carried out and the attained results are described.

Chapter 3

Test Cases

This chapter is concerned with test cases. To investigate whether the linear system of equations in PARNASSOS could better be solved with the MRILU method or with preconditioned GMRES, the MRILU factorization has been applied to the entire system $\boldsymbol{Ax} = \boldsymbol{b}$ and to each of the g block systems, after which the gained results have been compared to that of the current method. The following four sections describe the test cases which have been carried out.

In the first section, the effect of grid refinement is investigated.
The next section studies the influence of an enlargement of the sub-domains.
At the back-yard of the ship, flow seperation takes place. This can have influence on the coefficient matrix A and with that the solvation of the linear system (1.3). Therefore, the tests were performed out at both the midship and the back-yard of the ship. This is described in section 3.3.
§3.4 shows the effect of an adjustment of two input parameters of MRILU.

In order to investigate the effect of these adjustments, the results are represented in a table. The table shows the effect of the adjustments on the time (sec.) for preconditioning and solving for both GMRES and MRILU.

3.1 Grid Refinement

Grid refinement has been achieved by a refinement of the grid in i- (main-stream) and k-(girthwise) direction simultaneously. In the present version of PARNASSOS, grid refinement has been applied by decreasing the value of *istep* which also leads to a decrease of *kstep*. In here, *istep* and *kstep* denote the number of steps between two grid-points in i-direction and k-direction, respectively. For example, on the finest grid, both *istep* and *kstep* equals one, which means that only one step in i- and k-direction is made before going to the next grid point. In other words, an increase in *istep* or *kstep* results in a coarser grid. The number of grid points in mainstream direction and girthwise direction is 137 and 37, respectively.

Next to the size of a sub-domain (thus the value of g), the size of the coefficient matrix $\boldsymbol{A}$ is determined by the number of steps in k-direction (thus the value *kstep*) as well.

The size N of matrix $\boldsymbol{A}$ is obtained by the following formula.

$$N = g \times (4 \times NY \times (1 + (NZ - 1)/kstep)) \tag{3.1}$$

Although, grid refinement is obtained with a reduction of both *istep* and *kstep*, it will, for clarity, be denoted by a change of *kstep* only.

For the present tests $g = 2$ and the grid has been halved twice ($kstep = 4, 2$ and 1). Tests have been carried out halfway the ship (midship).
The effect of grid refinement on the preconditioning time and the time for solving for both MRILU and GMRES is represented in Table 3.1.

Grid Refinement												
		MRILU				**MRILU (GS)**			**Prec. GMRES**			
g=2, midship	N	Time Prec	Time Solve	Tot Time	gmres it.	Time Prec	Time Solve	Tot Tot	Time Prec	Time Solve	Tot Time	gmres it.
$kstep = 4$	6480	0.54	0.02	**0.56**	5	0.26	0.07	**0.33**	0.03	0.20	**0.23**	24
$kstep = 2$	12312	1.29	0.05	**1.34**	6	0.46	0.35	**0.81**	0.07	0.47	**0.54**	29
$kstep = 1$	23976	3.95	0.18	**4.13**	8	1.00	0.74	**1.74**	0.13	0.96	**1.09**	30

Table 3.1: *Results of a refinement of the mesh size.*

The following can be concluded from Table 3.1.
First of all, one could notice that, as a result of equation (3.1), grid refinement leads to an increase of the coefficient matrix $\boldsymbol{A}$. Together with an increase of $\boldsymbol{A}$, an increase in time for preconditioning and solving for both MRILU and precondtioned GMRES appears. If the grid is halved, these times are doubled.

Also the number of GMRES iteration steps that has been made is shown. These are the number of steps that is needed to achieve a 2-norm[8] of the residual smaller than 10^{-7}, i.e. $||\boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}|| < 10^{-7}$. This is quite an accurate requirement. In the current implementation a 2-norm of 10^{-2} has been taken, which is sufficient enough to achieve a decent solution. However, to make a comparison between MRILU and GMRES, high accuracy is needed. More about this in the next chapter.
The number of GMRES iteration steps within the MRILU method does not show a remarkable increase. Due to the accurate preconditioning, little steps are needed to to let the iteration converge.
What strikes is the difference between the two approaches of the MRILU method. The times that are needed for preconditioning and solving in the case of the approach in which MRILU has been applied to the entire system of linear equations, are quite a bit longer than in the case in which MRILU has been combined with the Gauss-Seidel loop.
The above is a result of small elements that occur during the factorization process. At every step of the Algorithm 1 (page 11), these small elements will end up in the Schur complement. As a result, the factorization of the next Schur complement will be much more difficult as if

these small elements had been omitted. If the system is large, $\boldsymbol{A}$ contains a lot of these small elements, such that the factorization is a quite time-consuming job. A somewhat artificial way to avoid this, is described in §3.4.2.
It has appeared that if the system is enlarged too much, this approach of MRILU will break down (§3.2).

Observing the cost for preconditioning and solving, one could notice that, for MRILU, the cost for the construction of the preconditioner is *larger* than the cost for solving. This is due to the accurate (time-consuming) factorization, which results in a solvation that is quite easier to make, i.e. the cost for solving will be less. For precondtioned GMRES, on the other hand, a less accurate preconditioning technique results in small preconditioning times and large solving times.
However, the most important conclusion that have to be made is that, for this case, one could better choose for the current method than for MRILU. The total time for both preconditioning and solving is larger for MRILU than for preconditioned GMRES. Maybe an enlargement of the sub-domains might help.

3.2 Large Sub-domains

As mentioned before, an increase of the sub-domains with a factor g results in a coefficient matrix, that is g times larger also (eq. 3.1). The effect of the enlargement on the time for preconditioning and for solving is represented in Table 3.2.
The tests have been carried out at the midship again, and *kstep* has been chosen equal to 4.

Only a comparison between the Gauss-Seidel (GS) block approach and the current version of the linear solver of PARNASSOS is shown, since the results achieved with MRILU applied on the entire system were useless, because of the enormous systems which have to be solved (§3.1).

VERGROTEN SUB-DOMEIN								
		MRILU (GS)			PARNASSOS			
kstep=4, midden	N	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	gmres it
g=4	6480	0.26	0.07	**0.33**	0.03	0.2	**0.23**	24
g=8	25920	1.55	0.31	**1.86**	0.14	2.54	**2.68**	52
g=17	12312	3.93	0.77	**4.7**	0.29	7.17	**7.46**	62
g=34	110160	12.14	2.43	**14.57**	0.58	17.33	**17.91**	70
g=50, kstep=2	307800	40.78	7.04	**47.82**	1.65	98.84	**100.49**	108

Table 3.2: *Results of an enlargement of the sub-domains.*

From Table 3.2 it becomes clear that for MRILU the time for preconditioning is still large,

compared to the time for solving. Preconditioned GMRES shows the opposite.
However, the sum of both times shows that if the size of a sub-domain becomes larger than 4, MRILU will be faster than preconditioned GMRES. If g is taken equal to 50 MRILU does not even use half the time that is needed to solve the linear system with preconditioned GMRES!

The conclusion could be made that the current method can handle large sub-domains badly. The cost for preconditioning is still small, however, the cost for solving increases awkwardly. On the contrary, MRILU (GS) works well in this situation.

In addition, in case of $g = 8$, grid refinement has been applied. The results are shown in the table below.

MIDDEN EN ACHTER											
		MRILU			MRILU (GS)			PARNASSOS			
g=8, midden	N	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	gmres it
kstep=4	25920	13.64	0.2	**13.84**	1.49	0.27	**1.76**	0.14	1.76	**1.9**	41
kstep=2	49248	x	x	**x**	2.91	0.48	**3.39**	0.23	3.92	**4.15**	46
kstep=1	95904	x	x	**x**	5.91	0.94	**6.85**	0.52	9.45	**9.97**	52

Table 3.3: *Results of grid refinement in the case of $g = 8$.*

On the whole, Table 3.3 shows the same behaviour as in case of sub-domains of size 2. There are two differences that can be noticed.
Firstly, as shown in the three columns at the left-hand side of Table 3.3, MRILU (without the GS-loop) cannot handle the enormous system (as a result of grid refinement) of equations anymore (§3.1).
And secondly, for this situation of sub-domains of size 8, MRILU will be faster than the current solution strategy. The total cost for both precondtioning and solvation of the linear system of equations is smaller if MRILU has been used, *even* if grid refinement is applied.

3.3 The Stern Area

At the back of the boat, in the stern area, the viscous effects on the wave pattern are present, which can be shown by the larger pressure differences at the wave surface. This is the region where flow seperation takes place. Now, the upper-triangular matrix not only contains elements coming form the discretization of the pressure derivative in the main-stream direction, but in this case of flow seperation, the matrix elements can also contain small distributions from the derivative of the velocities in main-stream direction.
For the case of $g = 8$, the effect of solving the linear system in the stern area, has been investigated. The result are represented in Table 3.4

MIDDEN EN ACHTER											
		MRILU			MRILU (GS)			PARNASSOS			
g=8, midden	N	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	gmres it
kstep=4	25920	13.64	0.2	**13.84**	1.49	0.27	**1.76**	0.14	1.76	**1.9**	41
kstep=2	49248	x	x	**x**	2.91	0.48	**3.39**	0.23	3.92	**4.15**	46
kstep=1	95904	x	x	**x**	5.91	0.94	**6.85**	0.52	9.45	**9.97**	52
		MRILU			MRILU (GS)			PARNASSOS			
g=8, zog	N	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd	gmres it
kstep=4	25920	16.07	0.23	**16.3**	1.47	0.38	**1.85**	0.13	2.48	**2.61**	52
kstep=2	49248	250	1.28	**251.28**	4.07	0.76	**4.83**	0.26	5.65	**5.91**	58
kstep=1	95904	x	x	**x**	11.67	1.88	**13.55**	0.53	16.32	**16.85**	74

Table 3.4: *Results of computations at the midship and back-yard of the ship.*

If the gained results are compared to the results in Table 3.3, the same bahaviour could be noticed. Although, the times for preconditioning and solving have been decreased. This a result of the small velocities in main-stream direction in the stern area. It is a result of the (second-order upwind) discretization of the convective terms

$$u(u_\xi^{n+1}) = u^n(\frac{u_i - u_{i-1}}{h_\xi}),$$

(where h_ξ denotes the step size in ξ-direction) in which the u_i have a small contribution to the main diagonal of the coefficient matrix $\boldsymbol{A}$, i.e. the speed of convergence will be decreased (see f.e. [7]).

3.4 Input Parameters for MRILU

As mentioned in §2.3, the essentiel ingredients of MRILU are the ordering and dropping. The ordering and dropping can be done in several ways. The dropping, performed in step 1 and 2 of the MRILU algorithm (on page 11), is based on a threshold parameter, the drop tolerance ϵ. The drop tolerance ϵ has great influence on the cost for the construction of the preconditioner and with that on the cost for solving [4]. This section shows wat happens when going from a rough preconditioner ($\epsilon = 0.5$) toward a accurate preconditioner ($\epsilon = 0.01$, where $\epsilon = 0$ corresponds to a complete factorization).

Next to the effect of the above, this section studies the effect of omitting small elements that occur in each Schur complement (see Algorithm 1 on page 11).

Both test cases have been carried out in the midship with $kstep = 4$.

3.4.1 The drop tolerance ϵ

Table 3.5 shows the cost for preconditioning, the cost for solving and the total cost for various values of ϵ.

DROP-TOLERANCE				
		MRILU		
g=8, midden	N	tijd prec	tijd solve	tot tijd
ε=0.5	25920	5.03	1.93	**6.96**
ε=0.1	25920	15.76	0.4	**15.8**
ε=0.01	25920	44.94	0.25	**45.19**

Table 3.5: *Results for various values of the drop tolerance ϵ.*

The behaviour that is shown in the table above, is typical for MRILU; a decrease of ϵ makes the cost for preconditioning expensive, while the cost for solving is cheap [4].
For the test cases described in this chapter, the drop tolerance has been taken equal to 0.1, i.e. a rather accurate preconditioner. This value has been chosen, since, in contrast to GMRES, the time for preconditioning is large. It turned out that an enlargement of ϵ did not result in a total time for preconditioning and solving smaller than that of the current implemented method.

3.4.2 Omitting small elements

The effect of omitting small elements during the construction of the MRILU preconditioner (§2.3.1) is represented in Table 3.6. The table shows what happens if MRILU is applied to the entire system of linear equations, as well as what happens when MRILU is combined with GS.

KLEINE ELEMENTEN							
		MRILU			MRILU (GS)		
g=8, midden, kstep=2	N	tijd prec	tijd solve	tot tijd	tijd prec	tijd solve	tot tijd
0	49248	x	x	**x**	2.75	0.43	**3.18**
<1E-7	49248	67.4	0.78	**68.18**	2.79	0.5	**3.29**
<1E-5	49248	42.58	0.57	**43.15**	2.52	0.14	**2.67**

Table 3.6: *The effect of omitting small elements during the construction of the MRILU preconditioner*

The first column shows the values that determine whether or not an element should be dropped. If a certain element in absolute value is smaller than the value in the table, it has been dropped.

The conclusion could be made that, if small elements are dropped, the MRILU method *can* be applied to the complete system of linear equations (2.1). The right-hand side of the table in which MRILU has been combined with GS does not show remarkable effects. Apparently, this approach is not influenced by the omission of the smalles elements, since they were already been moved.
However, the above described method is of course only an artificial way of solving the problem of the occurance of small elements.
Still, the improvements that have been achieved by omitting them, may lead to suggestions for future work. The next chapter gives an overview of the gained results, together with a list of suggestions for future research.

Chapter 4

Conclusions and Future Work

(Ik ben nog bezig dit hoofdstuk te veranderen. Hieronder alvast een puntsgewijs de inhoud.)

In this thesis, the MRILU-factorization, as a preconditioner for the system of linear equations in PARNASSOS, has been considered. It is a method in which the ordering of the unknowns and the dropping are determined during the factorization. Since it had succesfully been applied to a number of problems, it has been tested for the situation in PARNASSOS.
The gained results have shown that for a number of cases, MRILU works well.

Below a list of conclusions, together with a few recommendations for future work.

Conclusions:

- For small sub-domains MRILU one could better use the current method; for large sub-domains ($g > 4$) MRILU is faster.
- The cost for the construction of the preconditioner is large.
  MRILU applied on the entire system breaks down for large systems.
- The drop tolerance ϵ can reduce the time for preconditioning.
- Negletting small elements during the factorization makes the contruction of the preconditioner quite easier; the time for preconditiong reduces. MRILU is able to handle the entirre system as a whole.
- MRILU is only better in the case of high accuracy.

Future Work:

- Decreasing the time for preconditioning.
  - Re-use of the preconditioner matrix.
  - Neglecting the upper diagonal elements during the construction of th preconditioner.
- A scaling of the coefficient matrix.

Appendix A

Theory

1. **Diagonal Dominant**
   A real $N \times N$ matrix $\boldsymbol{A}$ is said to be diagonally dominant if

$$a_{ii} \geq \sum_{j=1_{j \neq i}}^{N} |a_{ij}| \quad for\ all\ i = 1, \cdots, N \tag{A.1}$$

   The matrix is said to be strictly diagonally dominant, wahen '$\geq$' is replaced by '$>$'.

2. **Flow Seperation**
   When the primary flow near the surface no longer continues to follow the surface contour because of sudden change in contour or because of insufficient flow momentum to allow flow to proceed into a region of increasing pressure.

3. **Hessenberg Matrix**
   A matrix of the form

$$\begin{bmatrix} a_{11} & a_{12} & a_{13} & \cdots & a_{1(n-1)} & a_{1n} \\ a_{21} & a_{22} & a_{23} & \cdots & a_{2(n-1)} & a_{2n} \\ 0 & a_{32} & a_{33} & \cdots & a_{3(n-1)} & a_{3n} \\ 0 & 0 & a_{43} & \cdots & a_{4(n-1)} & a_{4n} \\ 0 & 0 & 0 & \cdots & a_{5(n-1)} & a_{5n} \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & a_{(n-1)(n-1)} & a_{(n-1)n} \\ 0 & 0 & 0 & 0 & a_{n(n-1)} & a_{nn} \end{bmatrix}.$$

   Figure A.1: *Hessenberg Matrix*

4. **Jacobian Matrix**
   Given a set $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$ of n equations in n variables $x_1, \cdots, x_n$, written explicitly as

$$\boldsymbol{y} = \begin{bmatrix} f_1(\boldsymbol{x}) \\ f_2(\boldsymbol{x}) \\ \vdots \\ f_n(\boldsymbol{x}) \end{bmatrix}, \tag{A.2}$$

or more explicitly as

$$\begin{cases} y_1 = f_1(\boldsymbol{x_1} \cdots \boldsymbol{x_n}) \\ y_2 = f_2(\boldsymbol{x_1} \cdots \boldsymbol{x_n}) \\ \vdots \\ y_n = f_n(\boldsymbol{x_1} \cdots \boldsymbol{x_n}), \end{cases} \tag{A.3}$$

the Jacobian matrix, sometimes simply called "the Jacobian" (Simon and Blume 1994) is defined by

$$\boldsymbol{J}(x_1 \cdots x_n) = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_n}{\partial x_1} & \cdots & \frac{\partial y_n}{\partial x_n} \end{bmatrix}. \tag{A.4}$$

5. **Krylov Subspace**
   The Krylov subspace of dimension k is the linear space defined as follows.

$$K_k(\boldsymbol{A}; \boldsymbol{r}^{(0)}) = span\{\boldsymbol{r}^{(0)}, \boldsymbol{A}\boldsymbol{r}^{(0)}, \boldsymbol{A}^2\boldsymbol{r}^{(0)}, \cdots, \boldsymbol{A}^{k-1}\boldsymbol{r}^{(0)}\} \tag{A.5}$$

6. **Linear System of Equations**
   A linear system of equations is a set of n linear equations in k variables Linear systems can be represented in matrix form as the matrix equation

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b} \tag{A.6}$$

   where $\boldsymbol{A}$ is the matrix of coefficients, $\boldsymbol{x}$ is the column vector of variables, and $\boldsymbol{b}$ is the column vector of solutions.

   If $k < n$, then the system is (in general) overdetermined and there is no solution.
   If $k = n$ and the matrix is nonsingular, then the system has a unique solution in the n variables. In particular, as shown by *Cramer's rule*, there is a unique solution if has a matrix inverse . In this case,

$$\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b} \tag{A.7}$$

   If $\boldsymbol{b} = 0$, then the solution is simply $\boldsymbol{x} = 0$. If $\boldsymbol{A}$ has no matrix inverse, then the solution subspace is either a line or the empty set.

7. M**-matrix**
   A real matrix $\boldsymbol{A}$ is an M-matrix, if it is non-singular, its entries $a_{ij} \leq 0$ for $i \neq j$, and all entries of A^{-1} are not negative.

8. **2-Norm**
   The *2-norm*, *L2-norm* or *Euclidean norm*
   gives the length of an n-vector

$$\boldsymbol{x} = (x_1 \cdots, x_n) \tag{A.8}$$

by

$$\sqrt{x_n^1 + \cdots x_n^2}. \tag{A.9}$$

(variously denoted as $||\boldsymbol{x}||_2$ or $|\boldsymbol{x}|$)

9. **Sparse Matrix**
   A matrix which has only a small number of nonzero elements.

10. **Spectrum of Eigenvalues**
    The eigenvalues of a matrix $\boldsymbol{A}$ are called its spectrum, and are denoted $\lambda(\boldsymbol{A})$.

11. **Condition Number**
    The condition number of a nonsingular matrix $\boldsymbol{A}$ with respect to a matrix norm is

$$||\boldsymbol{A}||\ ||\boldsymbol{A}^{-1}||.$$

    The condition number is basically a measure of stability or sensitivity of a matrix (or the linear system it represents) to numerical operations. In other words, we may not be able to trust the results of computations on an ill-conditioned matrix.

    Matrices with condition numbers near 1 are said to be well-conditioned. Matrices with condition numbers much greater than one are said to be ill-conditioned.

Bibliography

[1] Ploeg, A. van der, Eça, L. and Hoekstra, M., *Combining accuracy and efficiency with robustness in ship stern flow computation*

[2] Hoekstra, M., *Numerical Simulation of Ship Stern Flows with a Space-Marching Navier-Stokes Method*, Master's thesis, Technische Universiteit Delft, 1999

[3] Ploeg, A. van der, *Preconditioning for Sparse Matrices with Applications*, Master's thesis, Rijksuniversiteit Groningen, 1994

[4] Botta, E.F.F. and Wubs, F.W., *Matrix Renumbering ILU: An Effective Algebraic Multi-level ILU Preconditioner for Sparse Matrices*, Siam J. Matrix Anal. Appl., Vol. 20, No. 4, pp. 1007-1026
(http://epubs.siam.org/sam-bin/getfile/SIMAX/articles/31930.pdf)

[5] Atkinson, Kendall E. *An Introduction to Numerical Analysis*, Second Edition, Canada, 1989

[6] Veldman, A.E.P., Numerieke Stromingsleer, Lecture notes, Rijksuniverstiteit Groningen, 1994

[7] Botta, E.F.F., Eindige-Differentiemethoden, Lecture notes, Rijksuniverstiteit Groningen, 1998

[8] A Portland Group Fortran 90 compiler manual.
(link: http://wwwuser.gwdg.de/ applsw/Parallelrechner/pghpf/man1/pgf90.html

[9] The web's most extensive mathematics resource.
(link: http://mathworld.wolfram.com/)

[10] The MRILU page of dr. ir. F.W. Wubs.
(link: http://www.math.rug.nl/ wubs/mrilu/)

[11] An online version of the master thesis of Dr. Ir. G. Tiesinga: *Multi-level ILU preconditioners and continuation methods in fluid dynamics*
(link http://www.ub.rug.nl/eldoc/dis/science/g.tiesinga/c4.pdf)

Photo Credit

Front Cover:

0