Chapter 2

Block preconditioners based on a repeated red-black ordering

2.1 Introduction

When solving flow problems often large sparse systems of equations, $Ax = b$, occur. Using a direct method to solve these systems results in large cpu-times and high computer storage demands. In such a method an LU-factorization of A is made, $A = LU$. The construction of L and U is very expensive when the dimension of the problem is large. Furthermore, the matrices L and U are in general not sparse, even when A is, demanding a high storage capacity. Therefore, iterative methods (e.g. a conjugate gradient type method) are often preferred. The convergence of such methods can be improved considerably by using a preconditioner P. The iterative method is applied to the preconditioned system $P^{-1}Ax = P^{-1}b$. An important class of preconditioners is formed by the (modified) incomplete LU-factorizations ((M)ILU), $A = LU + R$ with R small in some sense. The quality of these preconditioners strongly depends on the ordering of the unknowns. A good ordering strategy reduces the amount of fill and work needed for the factorization and increases the convergence rate of the iterative method.

Axelsson and Eijkhout [3] and Brand [9] introduced a preconditioning method based on a (modified) incomplete factorization of A with respect to a repeated red-black (RRB) ordering. Notay [35] showed that this method is of nearly optimal order, i.e. almost grid independent convergence, for second order discrete elliptic PDEs. In this chapter we will introduce a block form of the RRB preconditioner (see also [46]) which can be applied when solving discrete systems of PDEs in a fully coupled way. The use of an RRB preconditioner is attractive because its construction is cheap and can be vectorized easily.

An example of a system of PDEs which we will consider in this chapter is given by the incompressible Navier-Stokes equations. A popular way of solving these equations is to use a pressure-correction method. In this method the interlinkage between the equations is rather weak, which can result in a deterioration of the convergence rate. Furthermore, such a method needs time or time-like steps even when a steady solution is computed. These drawbacks can be avoided when the equations are solved fully coupled, i.e. all discrete equations are combined in one system which is then solved as a whole. In this approach the linear systems to be solved are very large and non-symmetric. However,

with the increase in computer storage, the presence of good iterative methods and the development of efficient preconditioners a coupled iterative approach has become feasible. For a comparison of both approaches see [52].

We will solve the Navier-Stokes equations fully coupled. After discretization of the equations with a finite difference method and linearization of the resulting discrete equations, large linear systems have to be solved. These systems can be written in block form in a natural way. The unknowns (velocity in x- and y-direction and pressure) as well as the equations (x- and y- momentum and continuity equation) per cell are grouped together, resulting in a system consisting of blocks of order three. Then, an iterative method combined with the block preconditioner based on an RRB ordering will be used to solve the systems.

The remainder of this chapter is organized as follows. The construction of the block preconditioner, a (modified) incomplete block factorization with respect to an RRB ordering is explained in Section 2.2. In Section 2.3 the discretization which we used in our flow problem calculations is explained with the Navier-Stokes equations. During the discretization a block form is introduced in a natural way. The results of numerical experiments are given in Section 2.4. Various forms of the block preconditioner are compared with each other. The influence of applying Gustafsson's modification, factorizing the last level of unknowns exactly and the number of levels in the RRB ordering on the efficiency of the preconditioner is studied. As test problems we have solved the Laplace equation (Section 2.4.1), the lid-driven cavity problem (Section 2.4.2) and two kinds of natural convection flows (Section 2.4.3). Finally, in Section 2.5 we give some conclusions.

2.2 Ordering and block factorization

Before constructing the preconditioner the cells are reordered using a repeated red-black (RRB) ordering. The cells are divided into several subsets according to the different steps in the RRB ordering. In the first step a conventional red-black ordering is imposed on the cells. The first subset is formed by the red cells. In the second step the remaining cells are also ordered red-black. These red cells form the second subset. This process can be repeated. An RRB ordering with three levels is shown in Figure 2.1 on a 9×9 grid.

The construction of the block preconditioner based on the RRB ordering is shown in stencil form. The preconditioner is an incomplete factorization of the matrix A. We assume the cells are connected to each other by a five-point block stencil z

$$\begin{array}{ccc} & N^{(0)} & \\ W^{(0)} & C^{(0)} & E^{(0)} \\ & S^{(0)} & \end{array}$$

The elements $S^{(0)}, W^{(0)}, C^{(0)}, E^{(0)}$ and $N^{(0)}$ of the stencil are blocks of small size. With the RRB ordering the cells in the first subset are not connected to each other. In the first step of the factorization these cells are eliminated. The remaining cells will be connected

$$\begin{array}{ccccccccc}
4 & 1 & 3 & 1 & 4 & 1 & 3 & 1 & 4 \\
1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\
3 & 1 & 4 & 1 & 3 & 1 & 4 & 1 & 3 \\
1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\
4 & 1 & 3 & 1 & 4 & 1 & 3 & 1 & 4 \\
1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\
3 & 1 & 4 & 1 & 3 & 1 & 4 & 1 & 3 \\
1 & 2 & 1 & 2 & 1 & 2 & 1 & 2 & 1 \\
4 & 1 & 3 & 1 & 4 & 1 & 3 & 1 & 4
\end{array}$$

Figure 2.1: A three step RRB ordering on a 9×9 grid, each cell is denoted by the index of the subset it belongs to.

by a skew nine-point block stencil

$$\begin{array}{ccccc}
 & & N^{(1)} & & \\
 & NW^{(1)} & & NE^{(1)} & \\
W^{(1)} & & C^{(1)} & & E^{(1)} \\
 & SW^{(1)} & & SE^{(1)} & \\
 & & S^{(1)} & &
\end{array}$$

When continuing the factorization, the skew nine-point block stencil in the cells of the second subset will be modified to a skew five-point block stencil

$$\begin{array}{ccc}
NW^{(1)} & & NE^{(1)} \\
 & \hat{C}^{(1)} & \\
SW^{(1)} & & SE^{(1)}
\end{array}$$

The stencils in the other cells are left unchanged. In this way the coupling between the cells of the second subset is removed. For the central block $\hat{C}^{(1)}$ we consider two options. The first option is to take $\hat{C}^{(1)} = C^{(1)}$, that is, the fill-in not corresponding to the skew five-point block stencil is dropped. The second option is to apply Gustafsson's modification, that is, the dropped fill-in is lumped on the diagonal in order to preserve the row-sum. For a block stencil this may be achieved by taking $\hat{C}^{(1)} = C^{(1)} + S^{(1)} + W^{(1)} + E^{(1)} + N^{(1)}$. Since fill-in is neglected this process will lead to an incomplete factorization. Eliminating the cells of the second subset results in a nine-point block stencil

$$\begin{array}{ccc}
NW^{(2)} & N^{(2)} & NE^{(2)} \\
W^{(2)} & C^{(2)} & E^{(2)} \\
SW^{(2)} & S^{(2)} & SE^{(2)}
\end{array}$$

in the remaining cells. In the third step of the factorization process the coupling between the cells in the third subset is removed by modifying in these cells the nine-point block

stencil to a five-point block stencil

$$\begin{array}{ccc} & N^{(2)} & \\ W^{(2)} & \hat{C}^{(2)} & E^{(2)} \\ & S^{(2)} & \end{array}$$

where $\hat{C}^{(2)} = C^{(2)}$ or $\hat{C}^{(2)} = C^{(2)} + SW^{(2)} + SE^{(2)} + NW^{(2)} + NE^{(2)}$. This is a similar kind of stencil as at the beginning of the factorization but on a double distance grid. Then these cells are eliminated, resulting again in a skew nine-point block stencil.

The elimination process is stopped after k steps. The remaining cells form the last subset. Two options are considered for this last subset. The first option is to replace the nine-point block stencil by its central block $C^{(k)}$, resulting in a one-point block stencil (the last part of the factorization is a diagonal). In this way an expensive factorization of the last level is avoided. The second option is to retain all fill-in and compute an exact factorization for this last set of unknowns.

2.3 The Navier-Stokes equations discretized in block form

An important system of PDEs occurring in fluid dynamics consists of the Navier-Stokes equations. We solve these equations fully coupled, instead of using a pressure-correction method. After discretization of these coupled equations the resulting discretized system can be written in block form in a natural way. Thus, enabling us to use the block preconditioner introduced in the previous section.

The steady two-dimensional incompressible Navier-Stokes equations are given by

$$\begin{aligned} u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} &= -\frac{\partial p}{\partial x} + \nu\,(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}), \\ u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} &= -\frac{\partial p}{\partial y} + \nu\,(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}), \\ \frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0, \end{aligned}$$

where u and v are the velocities in x- and y-direction respectively, p the pressure and ν the viscosity of the fluid.

The equations are discretized on an equidistant staggered grid (see Figure 2.2) using a finite-difference formulation. The convective terms are discretized with a first-order upwind scheme. In cell (i,j) the x-momentum equation is discretized in position $(i, j+\frac{1}{2})$. The term $v\frac{\partial u}{\partial y}$ is discretized as follows

$$(v\frac{\partial u}{\partial y})_{i,j+\frac{1}{2}} = \begin{cases} v_{i,j+\frac{1}{2}}\dfrac{u_{i,j+\frac{1}{2}} - u_{i-1,j+\frac{1}{2}}}{h} & \text{if } v_{i,j+\frac{1}{2}} \geq 0, \\ v_{i,j+\frac{1}{2}}\dfrac{u_{i+1,j+\frac{1}{2}} - u_{i,j+\frac{1}{2}}}{h} & \text{if } v_{i,j+\frac{1}{2}} < 0, \end{cases}$$

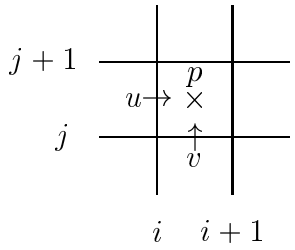

Figure 2.2: The positions of u,v and p in cell (i, j).

with h the grid size. The velocity in y-direction v is not defined at $(i, j+\frac{1}{2})$. Therefore, in this position, v is approximated by the average of the surrounding velocities in y-direction

$$v_{i,j+\frac{1}{2}} = \frac{1}{4}(v_{i+\frac{1}{2},j} + v_{i-\frac{1}{2},j} + v_{i+\frac{1}{2},j+1} + v_{i-\frac{1}{2},j+1}).$$

The diffusive terms are discretized using the standard second-order central scheme.

Because the equations are solved fully coupled, the unknowns u, v and p of each cell are put in a state vector x

$$x = (\ldots, u_{i,j+\frac{1}{2}}, v_{i+\frac{1}{2},j}, p_{i+\frac{1}{2},j+\frac{1}{2}}, u_{i+1,j+\frac{1}{2}}, v_{i+1\frac{1}{2},j}, p_{i+1\frac{1}{2},j+\frac{1}{2}}, \ldots).$$

The discrete system is written as

$$f(x) = (f_1(x), f_2(x), \ldots, f_{3N}(x)) = 0,$$

with N the number of grid cells. The components $f_{3(l-1)+1}, f_{3(l-1)+2}$ and $f_{3(l-1)+3}$ for $l = 1, 2, \ldots, N$ of f are the discretized forms of the x-momentum, y-momentum, and continuity equation, respectively, at the l-th cell.

In a natural way the discretized system can be written in block form. In each cell the equations are grouped together and the unknowns u, v and p are joined in a vector variable

$$F_l = (f_{3(l-1)+1}, f_{3(l-1)+2}, f_{3(l-1)+3}),$$
$$w_{i,j} = (u_{i,j+\frac{1}{2}}, v_{i+\frac{1}{2},j}, p_{i+\frac{1}{2},j+\frac{1}{2}}).$$

In block form the discrete system is given by

$$f(x) = (F_1(x), F_2(x), \ldots, F_N(x)) = 0,$$

with $x = (\ldots, w_{i,j}, w_{i+1,j}, \ldots)$.

This system is nonlinear and is solved with Newton's method. The k-th step of this method is given by

$$\begin{aligned} J(x^{(k)})\delta &= -f(x^{(k)}), \\ x^{(k+1)} &= x^{(k)} + \delta, \end{aligned} \tag{2.1}$$

with J the Jacobian of the system. As a consequence of grouping the equations and unknowns per cell the elements of the Jacobian are 3×3 blocks. The group of equations belonging to cell (i,j) can be written as

$$S\ \delta_{i,j-1} + SE\ \delta_{i+1,j-1} + W\ \delta_{i-1,j} + C\ \delta_{i,j} + E\ \delta_{i-1,j+1} + NW\ \delta_{i-1,j+1} + N\ \delta_{i,j+1} = r_{i,j},$$
$$w_{i,j}^{(k+1)} = w_{i,j}^{(k)} + \delta_{i,j},$$

where S, SE, W, C, E, NW and N are 3×3 matrices. Hence, in stencil form the Jacobian is given by

$$\begin{array}{ccc} NW & N & \\ W & C & E \\ & S & SE \end{array}$$

The blocks NW and SE occur because we use the full Jacobian. These blocks stem from the linearization of the convective terms, because the velocity $v_{i,j+\frac{1}{2}}$ is approximated by an average of surrounding velocities in the discretization of the term $(v\frac{\partial u}{\partial y})_{i,j+\frac{1}{2}}$.

The block preconditioner introduced in the previous section has as starting point a system with a five-point block stencil. To apply this preconditioner here, the seven-point block stencil has to be modified to a five-point block stencil. Therefore, after imposing the repeated red-black ordering, the seven-point block stencil in the cells of the first subset is modified to a five-point block stencil

$$\begin{array}{ccc} & N & \\ W & \hat{C} & E \\ & S & \end{array}$$

with $\hat{C} = C$ or $\hat{C} = C + NW + SE$. In the other cells the stencils are unchanged. Now, the factorization can be performed as described in the previous section.

2.4 Numerical experiments

In this section, we compare the different variants of the block preconditioner based on an RRB ordering. These variants have been tested on several flow problems. In all test cases the effect of

1. the number of levels in the RRB ordering,
2. the use of Gustafsson's modification,
3. an exact factorization of the last part of the incomplete factorization (corresponding to the last set of cells)

on the performance of the preconditioner is tested.

In the first test case the Laplace equation is solved. First, this equation is solved using the block preconditioner with block size one, which we will denote by point preconditioner. Then, the equation is written as a system of three equations, corresponding to

the treatment of the pressure terms in the (Navier-)Stokes equations, and solved using the block preconditioner with block size three. The performance of the point and block preconditioner will be compared. As a second test case the flow in a lid-driven cavity, described by the two-dimensional steady incompressible Navier-Stokes equations, is computed. In the third and fourth test case the application of the block preconditioner is extended to a system of four partial differential equations: two kinds of natural convection flow, described by the Boussinesq equations (i.e. Navier-Stokes plus heat transfer), are considered.

In all experiments the outer iteration process (i.e. Newton's method) is stopped when the maximum norm of the difference between two succeeding iteration steps is less than 10^{-6},

$$\| x^{(k)} - x^{(k-1)} \|_\infty \leq 10^{-6}.$$

For the inner iteration the BiCGstab(2) method [43] has been used as linear solver. As stopping criterion for this method the Euclidean norm of the preconditioned residual is required to decrease with a factor 10^{-4},

$$\| P^{-1}(Ax_i^{(k)} - b) \|_2 \leq 10^{-4} \| P^{-1}(Ax_0^{(k)} - b) \|_2,$$

with i the index of the inner iteration and P the preconditioner. All calculations have been performed on an HP PA-8000 C160 workstation. The results are shown in the next subsections.

2.4.1 The Laplace equation

Preconditioners based on a repeated red-black ordering have already been studied extensively [3, 9]. Notay [35] showed recently that, when applied to discrete second order elliptic PDEs with isotropic coefficients, point preconditioners based on a modified ILU decomposition with respect to a repeated red-black ordering in which an exact factorization is used for the last level of unknowns are optimal, i.e. grid independent, in the number of iterations when the number of levels is kept fixed during grid refinement and nearly optimal when the number of levels is $\log_2 h^{-1}$ (with h the grid size). We consider block forms of such preconditioners. To see if the convergence behaviour of the point preconditioner based on an RRB ordering is preserved when it is extended to a block form the Laplace equation is solved using both the point and block preconditioner.

The Laplace equation is solved on the unit square with Neumann boundary conditions

$$\begin{aligned} \Delta p &= 0 \qquad \text{on} \quad \Omega = [0,1] \times [0,1], \\ \frac{\partial p}{\partial n} &= 0 \qquad \text{on} \quad \delta\Omega. \end{aligned} \tag{2.2}$$

This problem arises when solving the Navier-Stokes equations with a pressure-correction method. In each time step the pressure has to be computed from a Poisson equation which is defined by a Laplace operator. The Laplace operator is discretized on a uniform grid using the standard second-order central scheme. The resulting linear system is solved using the CG method with the point preconditioner with respect to a RRB ordering.

As an alternative the Laplace equation is written in system form

$$\left.\begin{aligned} u &= \frac{\partial p}{\partial x} \\ v &= \frac{\partial p}{\partial y} \\ \frac{\partial u}{\partial x} &+ \frac{\partial v}{\partial y} = 0 \end{aligned}\right\} \quad \text{on} \quad \Omega = [0,1] \times [0,1], \tag{2.3}$$

$$u = v = 0 \qquad \text{on} \quad \delta\Omega.$$

This system is discretized on a uniform grid as described in Section 2.3 and solved with the BiCGstab(2) method with the block preconditioner. The third equation of (2.3) is independent of p. As a consequence the matrix that occurs after discretization has zeros on the diagonal. Hence, without pivoting the construction of a point preconditioner may fail. By using a block preconditioner this can be prevented. Although some diagonal elements are zero, the diagonal blocks are non-singular. Herewith, we circumvent the use of a pivoting strategy which may easily destroy the structure of the matrix. As starting vector the function

$$p(x,y) = (xy(1-x)(1-y))^2 \mathrm{e}^{x^2 y}$$

evaluated at the grid points is used. When solving system (2.3) the extra starting vectors u and v are obtained by differentiating $p(x,y)$ to obtain $u = p_x$ and $v = p_y$, and evaluating them at the grid points. Various forms of the point and block preconditioner, using Gustafsson's modification (MILU) or not (ILU) and factorizing the last level exactly or not, will be compared. The results for the point preconditioner and the block preconditioner are given in Table 2.1 and 2.2 respectively.

This test case has been chosen to compare the convergence behaviour when using a block preconditioner and when using a point preconditioner. Therefore, we compare Table 2.1 and 2.2 with respect to the number of matrix-vector multiplications. The number of matrix-vector multiplications is a measure for the number of iterations of the preconditioned BiCGstab(2) method. For each variant of the preconditioner a similar qualitative convergence behaviour for the block and the point preconditioner is observed. We conclude that introducing a block form of the RRB preconditioner does not change the convergence behaviour.

The amount of work for one step of the solver is higher when using a block preconditioner than when using a point preconditioner. The number of unknowns of the Laplace system is three times the number of unknowns of the Laplace equation. Furthermore, using the block preconditioner requires operations with blocks of order three, while using the point preconditioner requires only scalar operations. Therefore, we see in Table 2.1 and 2.2 that when using a block preconditioner the cpu-time is about an order larger than when using a point preconditioner.

The differences in the performances of the variants of the preconditioner, i.e. the influence of Gustafsson's modification and an exact factorization of the last level, will be discussed now. Because the point and block preconditioner behave similarly, we will only consider the block variant.

		33×33 grid		65×65 grid		129×129 grid	
features of preconditioner	levels	mat-vec mult.	cpu-time	mat-vec mult.	cpu-time	mat-vec mult.	cpu-time
ILU	2	62	0.42	127	3.8	254	31.5
last level not exact	4	42	0.36	82	2.9	164	24.8
	6	41	0.37	79	2.9	157	25.0
ILU	2	38	0.50	72	5.6	141	69.3
last level exact	4	41	0.38	80	3.3	157	30.9
	6	41	0.38	78	3.1	157	26.9
MILU	2	67	0.43	134	4.1	267	33.0
last level not exact	4	35	0.31	66	2.4	127	20.2
	6	24	0.23	38	1.5	69	11.2
MILU	2	11	0.18	10	1.4	9	12.7
last level exact	4	16	0.17	16	0.8	15	4.0
	6	21	0.21	21	1.0	19	3.7

Table 2.1: Results of solving the Laplace equation (2.2) with a point preconditioner.

		33×33 grid		65×65 grid		129×129 grid	
features of preconditioner	levels	mat-vec mult.	cpu-time	mat-vec mult.	cpu-time	mat-vec mult.	cpu-time
ILU	3	77	4.6	137	36.4	257	303.6
last level not exact	5	61	4.1	117	33.5	185	240.5
	7	61	4.3	109	32.4	189	248.6
ILU	3	57	4.2	101	38.2	169	397.1
last level exact	5	61	4.5	109	32.1	189	256.1
	7	61	4.8	113	32.9	181	260.5
MILU	3	77	4.5	149	39.6	273	321.1
last level not exact	5	49	3.4	81	24.0	141	191.4
	7	49	3.3	61	19.3	85	116.6
MILU	3	25	2.7	29	19.3	29	221.2
last level exact	5	33	2.9	33	13.1	33	64.3
	7	37	3.2	40	15.2	41	66.8

Table 2.2: Results of solving the Laplace system (2.3) with a block preconditioner.

Last level exact First, we will look at the preconditioners in which the last part is factorized exactly. When the number of levels is increased, a smaller part is factorized exactly. This results in a loss of convergence. Nevertheless, the number of iterations will always be smaller than when the last part is not factorized exactly.

The cpu-time needed for the exact factorization of the last block will decrease when more levels are used. But, the cpu-time needed for the (M)ILU factorization of the first part will increase. Constructing an exact factorization is more expensive than constructing an incomplete factorization. Therefore, increasing the number of levels will reduce the cpu-time needed for both the factorization and one iteration. However, the total number of iterations will increase when the number of levels is increased. Therefore, for the complete solve one expects an optimum in the cpu-time. From the results it is seen that the cpu-time is minimal for 5 levels.

Last level not exact Secondly, we will consider the preconditioners in which the last part of the factorization, corresponding to the last set of unknowns, is replaced by a block diagonal matrix. The loss of convergence due to this approximation can be kept small if this last level is small. But, this requires more levels in the factorization, which will in itself give a loss of convergence due to the approximations at each level. Hence, one expects an optimum in the number of levels. This is observed in the results for the ILU factorization with the last level not exact; the increase from 5 to 7 levels has little effect. Clearly, the error of the ILU factorization at the first levels soon dominates the error of the diagonal preconditioning at the last level. From the results it is seen that for an MILU factorization with the last level not exact this optimum will be reached when more than 7 levels are used. The error in the MILU factorization is smaller than in the ILU factorization, and hence the error at the last level will dominate at higher levels as well.

For the MILU factorization the stencil at the last level is also an approximation to the Laplacian in block form, but on a coarser grid. When the number of levels is increased with two, this grid is twice as coarse, yielding at the diagonal preconditioned last part a reduction of the condition number by a factor 4. This reduction would halve the number of CG iterations. On the other hand, the number of iterations is also influenced by the MILU factorizations at the other levels. On the finest grid (last column in Table 2.2) the behaviour is clearly dominated by the approximation at the last level. This situation changes for coarser grids where we only see a reduction due to the approximation at the last level when going from 3 to 5 levels.

The cpu-time needed for the construction of the factorization and for one iteration step will increase only mildly when the number of levels is increased. Hence, the cpu-time is almost linear in the number of iterations, as can be observed from the results.

Grid refinement Finally, we will look at the convergence behaviour when the grid is refined. For a symmetric problem Gustafsson [22] showed that using a modified ILU(0) factorization improves the condition number from $O(N)$ to $O(N^{\frac{1}{2}})$, where N is the total number of unknowns. This means that the number of iterations with an ILU(0) factorization is $O(N^{\frac{1}{2}})$, and with an MILU(0) factorization $O(N^{\frac{1}{4}})$. For the MILU factorization with respect to a repeated red-black ordering this is a worst case estimate. Notay [35]

showed that when this preconditioner is applied to discrete second order elliptic PDEs the condition number is $O(N^{0.153})$. If at the last level diagonal preconditioning is used, the condition number at this last level is $O(N)$, and hence the number of iterations is $O(N^{\frac{1}{2}})$.

First, we will look in Table 2.2 at the diagonal from top left to bottom right. Along these diagonals the last blocks are of equal order, and, in case of the Laplace equation, the condition numbers of the diagonal preconditioned parts are equal. Therefore, the observed loss of convergence is caused by the factorizations of the first parts. Indeed, the convergence is majorized by the $O(N^{\frac{1}{2}})$ behaviour of ILU(0) and the $O(N^{\frac{1}{4}})$ behaviour of MILU(0).

Now, we will look at the rows of Table 2.2. The convergence with an ILU factorization with an exact factorization of the last level shows an $O(N^{\frac{1}{2}})$ behaviour, the number of iterations increases with a factor 2 when the number of unknowns is increased with 4. The convergence with an ILU factorization with a diagonal preconditioning at the last level is $O(N^{\frac{1}{2}})$ as well, because both an ILU factorization and a diagonal preconditioning have $O(N^{\frac{1}{2}})$ convergence. The MILU preconditioner with an exact factorization of the last level shows an almost grid independent convergence behaviour, this behaviour is majorized by $O(N^{\frac{1}{4}})$. With an MILU factorization with the last level not exact the convergence behaviour is $O(N^{\frac{1}{2}})$, which is a consequence of the $O(N^{\frac{1}{2}})$ convergence of the diagonal preconditioning of the last part.

Summarizing, we see that the convergence behaviour of the block preconditioner is the same as that of the point preconditioner; introducing a block form has no negative effect. Preconditioners based on MILU factorizations are more accurate than those based on ILU factorizations. The best results are obtained with the MILU factorization in which the last part is factorized exactly. The BiCGstab(2) method with this preconditioner shows an almost grid independent convergence.

2.4.2 Lid-driven cavity flow

A well-known problem to test solvers for the incompressible Navier-Stokes equations is the lid-driven cavity problem. This problem consists of calculating the flow in a square cavity with a constantly moving lid. The governing equations and the discretization method are given in Section 2.3. The domain and boundary conditions of the lid-driven cavity problem are shown in Figure 2.3.

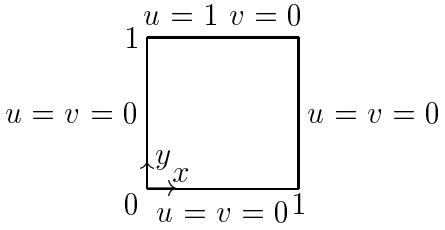

Figure 2.3: Geometry for the lid-driven cavity problem.

To compare the performances of the different forms of the block preconditioner, the flow fields for Reynolds numbers 100 and 1000 have been computed. The results are given in Table 2.3 and Table 2.4. On some points these results differ from the results of solving the Laplace system given in Section 2.4.1. We will consider the respective variants of the block preconditioner and point out the differences.

Last level exact First, the preconditioners with an exact factorization of the last part are considered. As in the case of solving the Laplace system, an increase of the number of levels in the RRB ordering results for both the ILU and the MILU preconditioner in an increase in the number of BiCGstab(2) iterations. When five levels are used in the ordering the MILU factorization causes a stagnation of the convergence of the BiCGstab(2) method. When the last part is factorized exactly the MILU factorization (except when five levels are used) gives better results than the ILU factorization.

Last level not exact Secondly, we consider the block preconditioner in which the last part is replaced by a block diagonal matrix. In the previous section we saw that for the Laplace system increasing the number of levels results in a decrease of the number of iterations needed for the BiCGstab(2) method. From Table 2.3 and 2.4 we see that for the lid-driven cavity problem both the MILU and the ILU factorization have this behaviour. However, with the MILU factorization based on five levels in the RRB ordering the convergence of the BiCGstab(2) method stagnates. For a Reynolds number equal to 1000 Gustafsson's modification does not have a positive effect: the MILU preconditioner performs in general worse than the ILU preconditioner.

Grid refinement Finally, we consider the convergence behaviour when the grid is refined. The preconditioner based on an MILU factorization in which the last part is factorized exactly is optimal (i.e. almost grid independent) for a fixed number of levels in the ordering (except when five levels are used). For the other variants of the preconditioner the number of iterations is increased with a factor of about 3 when the number of unknowns is increased by a factor 4.

Summarizing, the fastest convergence when solving the lid-driven cavity problem is obtained with a preconditioner based on an MILU factorization with respect to an RRB ordering with three levels and using an exact factorization of the last level of unknowns. The behaviour of the block preconditioner applied to the Navier-Stokes equations is generally the same as when applied to the Laplace system. However, the MILU variants with five levels in the ordering cause a stagnation in the convergence rate. This can be repaired by relaxing the modification (see [4]). In the next chapter we will explain the convergence behaviour when the MILU preconditioner with an exact factorization of the last part is used by looking at the eigenvalues of the preconditioned system. For most variants increasing the number of unknowns with a factor 4 results in an increase of the BiCGstab(2) iterations with a factor 3. The preconditioner is less efficient than for the Laplace system. This is not surprising, since the Navier-Stokes equations are severely non-symmetric, hence solving this system is more difficult than solving the Laplace system.

		33×33 grid			65×65 grid		
features of preconditioner	levels	Newton it.	mat-vec mult.	cpu-time	Newton it.	mat-vec mult.	cpu-time
ILU	1	6	2746	108	7	16075	3018
last level not exact	3	6	778	46	7	2991	794
	5	6	482	33	6	1306	384
ILU	1	6	30	51	6	30	807
last level exact	3	6	370	30	6	906	337
	5	6	442	30	6	1230	366
MILU	1	6	2914	114	7	12391	2292
last level not exact	3	7	727	45	7	2451	643
	5	-	-	-	-	-	-
MILU	1	6	30	51	6	30	794
last level exact	3	6	94	13	6	102	108
	5	7	2599	161	-	-	-

Table 2.3: Results of the lid-driven cavity problem for Re = 100, - denotes a stagnation in the preconditioned BiCGstab(2) process.

		33×33 grid			65×65 grid		
features of preconditioner	levels	Newton it.	mat-vec mult.	cpu-time	Newton it.	mat-vec mult.	cpu-time
ILU	1	9	4361	175	9	27665	5266
last level not exact	3	9	1305	76	8	3596	956
	5	9	797	53	9	2289	671
ILU	1	9	77	76	8	64	1038
last level exact	3	9	625	49	9	1533	569
	5	9	769	52	9	2169	646
MILU	1	9	4857	191	9	26157	4862
last level not exact	3	9	2017	116	9	6105	1594
	5	-	-	-	-	-	-
MILU	1	9	49	74	8	44	1018
last level exact	3	9	433	38	9	309	204
	5	10	8410	509	-	-	-

Table 2.4: Results of the lid-driven cavity problem for Re = 1000, - denotes a stagnation in the preconditioned BiCGstab(2) process.

2.4.3 Natural-convection flow

So far we have solved problems described by a system of three PDEs, in this section we will consider systems of four PDEs. We will calculate natural-convection flows [25] resulting from temperature differences. These flows are described by the Boussinesq equations (the Navier-Stokes equations plus a temperature equation):

$$\begin{aligned}
&\frac{1}{\mathrm{Pr}}(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}) = -\frac{\partial p}{\partial x} + (\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}),\\
&\frac{1}{\mathrm{Pr}}(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}) = -\frac{\partial p}{\partial y} + (\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}) + \mathrm{Ra}\ T,\\
&\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0,\\
&u\frac{\partial T}{\partial x} + v\frac{\partial T}{\partial y} = (\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2}),
\end{aligned}$$

where T is the temperature, Pr the Prandtl number and Ra the Rayleigh number. The equations are discretized on a staggered grid with T defined in the centers of the cells, again a first-order upwind scheme for the convective terms, as described in Section 2.3 for the Navier-Stokes equations, has been used. The variables per cell are grouped together, forming a vector variable of size four, $w_{i,j} = (u_{i,j+\frac{1}{2}}, v_{i+\frac{1}{2},j}, p_{i+\frac{1}{2},j+\frac{1}{2}}, T_{i+\frac{1}{2},j+\frac{1}{2}})$. The construction of the preconditioner is similar as described in Section 2.2, except that the blocks are now 4×4 instead of 3×3.

The natural-convection flows can be divided into two classes, characterized by the cause of the temperature differences: heating from a horizontal wall or heating from a vertical wall. First we consider Rayleigh-Bénard flows, these belong to the first class. Then results for calculations on a problem of the second class will be given.

Heating from a horizontal wall: Rayleigh-Bénard flow

As test case we consider Rayleigh-Bénard flows; a liquid in a rectangular box of aspect ratio A is heated from below. The domain and boundary conditions are given in Figure 2.4.

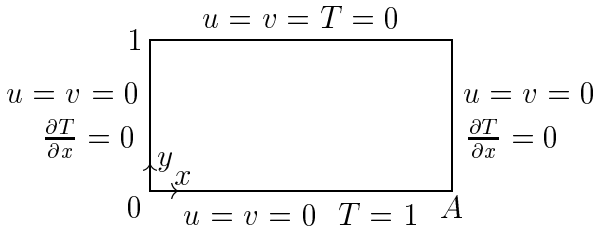

Figure 2.4: Geometry for the Rayleigh-Bénard problem.

Depending on the Rayleigh and Prandtl number different stable and unstable flow patterns (motionless flow, ten-cell pattern, nine-cell pattern and an eleven-cell pattern) can occur [15]. We will show results of calculations of the flow for Ra = 1750 and Pr = 5.5 in a

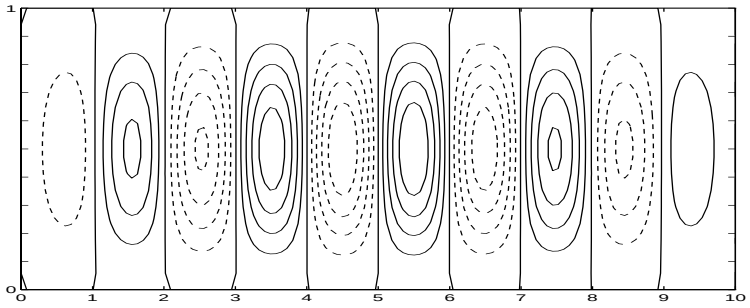

Figure 2.5: Streamlines of the 10-cell solution of the Rayleigh-Bénard problem for Ra = 1750 and Pr = 5.5.

features of preconditioner	levels	Newton it.	mat-vec mult.	cpu-time
ILU	1	6	12638	1986
last level not exact	3	6	2986	630
	5	6	1842	415
ILU	1	6	30	67
last level exact	3	6	1358	309
	5	6	1802	401
MILU	1	6	14542	2293
last level not exact	3	7	2739	608
	5	-	-	-
MILU	1	6	30	67
last level exact	3	6	2174	490
	5	-	-	-
direct solver		6		87

Table 2.5: Results of the Rayleigh-Bénard problem on a 129 × 17 grid with Ra = 1750 and Pr = 5.5, - denotes a stagnation in the BiCGstab(2) process.

box with aspect ratio $A = 10$. The computations are performed on an 129 × 17 grid. For these parameters we obtain the stable ten-cell pattern as shown in Figure 2.5.

In Table 2.5 the results for the various forms of the preconditioner are given. In this table the results of a direct solver are given as well. In most calculations a direct solver demands too much computer storage and cpu-time to be competitive with a preconditioned iterative method. However, in this test case the direct solver is competitive. This is a consequence of the small number of grid cells in the y-direction: the cells can be ordered in such a way that the matrix of the discretized system has a small bandwidth and hence can be factorized fast and with a low storage demand.

The convergence behaviour of the different variants of the preconditioner is similar to that in the lid-driven cavity calculations. The performance of the preconditioner does not change when the block size is increased to four. For this test case the only difference observed is with respect to the cpu-time. When an exact factorization is made of the

last part the least cpu-time is needed when one level is used in the ordering. This is a consequence of the small bandwidth of the Jacobian matrix, which allows a fast exact factorization.

Heating from a vertical wall

In the previous test case the direct solver is competitive with the preconditioned iterative method. To make sure this is only a consequence of the small bandwidth of the Jacobian matrix and is not caused by the system itself, we consider another natural-convection flow. This flow is induced by heating the left wall of a square. The geometry and boundary conditions are given in Figure 2.6.

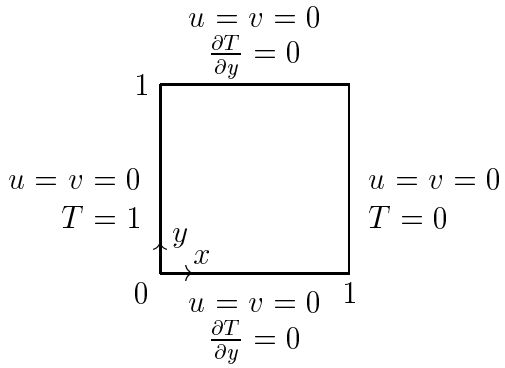

Figure 2.6: Geometry for the left-side heated natural-convection flow.

We determined solutions for a cavity filled with air (Pr = 0.71). For the Rayleigh numbers $10^4, 10^5$ and 10^6 the streamlines are given in Figure 2.7. For the lowest Rayleigh number the flow has one center, for Ra = 10^5 this center has split in a saddle point and two centers, and for Ra = 10^6 this saddle has further split up in two saddles and a center.

Again we have solved the problem using various forms of the block preconditioner. The results for Ra = 10^4 and Ra = 10^6 are given in Table 2.6 and 2.7, respectively. The results for Ra = 10^5 are similar and therefore not included.

The convergence rate of the preconditioned BiCGstab(2) method with the different variants of the factorization is the same as for the lid-driven cavity calculations. Again the MILU decomposition with respect to five levels in the ordering results in a stagnation in the BiCGstab(2) method. Furthermore, for Rayleigh number equal to 10^6 the MILU factorization in which the last part is replaced by a block diagonal matrix is less accurate than this variant of the ILU factorization. The best performance is obtained with the block preconditioner based on an MILU decomposition with respect to three levels of red-black ordering and an exact factorization of the last level of unknowns. When the number of levels in the ordering is fixed, the performance of the MILU factorization with an exact factorization of the last level of unknowns is grid independent, with the other preconditioners the number of iterations increases with a factor three when refining the grid with a factor two. In contrast to the Rayleigh-Bénard calculations one level in the RRB ordering combined with an exact factorization of the last part does not result in a low cpu-time. The direct solver is not competitive anymore as a consequence of the large number of grid cells in both directions.

		33×33 grid			65×65 grid		
features of preconditioner	levels	Newton it.	mat-vec mult.	cpu-time	Newton it.	mat-vec mult.	cpu-time
ILU	1	7	4623	648	7	19123	10205
last level not exact	3	7	1239	226	7	3595	3062
	5	6	638	130	7	2179	1764
ILU	1	6	30	226	6	30	3651
last level exact	3	7	539	151	7	1427	1848
	5	6	586	125	7	1827	1544
MILU	1	7	4351	576	7	17055	9427
last level not exact	3	7	947	179	7	3047	2611
	5	7	14555	3154	-	-	-
MILU	1	6	30	236	6	30	3743
last level exact	3	6	102	54	7	123	486
	5	7	1739	360	-	-	-

Table 2.6: Results for Ra = 10^4 and Pr = 0.71, - denotes a very slow convergence in the BiCGstab(2) process.

		33×33 grid			65×65 grid		
features of preconditioner	levels	Newton it.	mat-vec mult.	cpu-time	Newton it.	mat-vec mult.	cpu-time
ILU	1	8	3812	540	8	35816	21709
last level not exact	3	8	1108	214	8	3760	2802
	5	8	728	155	8	1920	1801
ILU	1	8	52	289	7	35	3962
last level exact	3	8	508	144	8	1408	1816
	5	8	692	153	8	1720	1479
MILU	1	8	4276	618	8	42264	25497
last level not exact	3	9	2377	462	8	3336	2457
	5	9	17237	3523	-	-	-
MILU	1	8	44	284	7	35	4043
last level exact	3	8	240	85	8	204	559
	5	10	3574	713	8	19268	17310

Table 2.7: Results for Ra = 10^6 and Pr = 0.71, - denotes a very slow convergence in the BiCGstab(2) process.

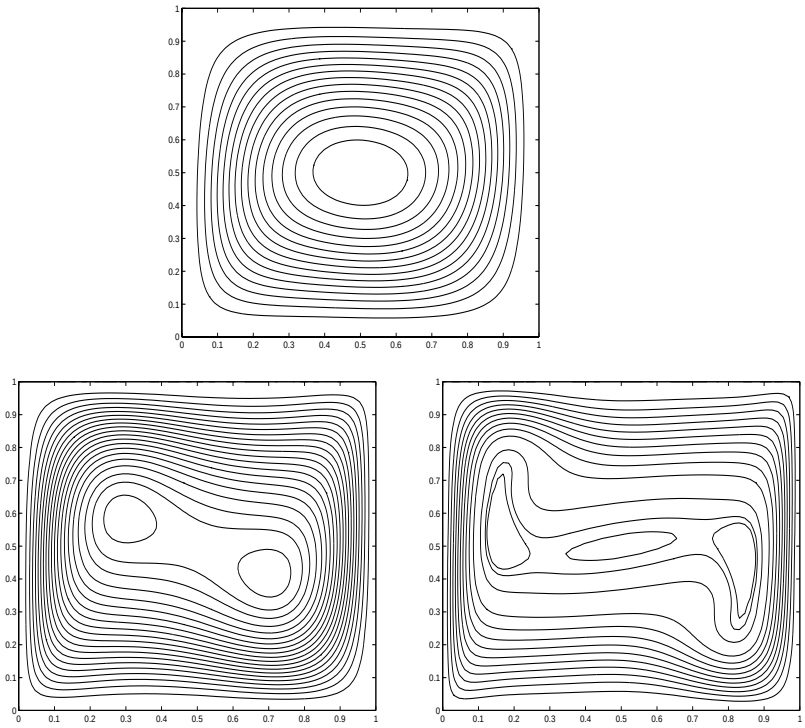

Figure 2.7: Streamlines of the natural-convection flows for air with Ra = 10^4, Ra = 10^5 and Ra = 10^6.

2.5 Conclusions

A preconditioned iterative solver for solving systems of PDEs has been developed. Preconditioners based on block incomplete factorizations with respect to a repeated red-black ordering of the grid cells were tested. The influence of Gustafsson's modification, the number of levels in the ordering, and the treatment of the last level of grid cells (an exact factorization or replacement by a block diagonal matrix) were investigated. As test cases flow problems described by the Laplace equation, Navier-Stokes equations and the Boussinesq equations have been considered.

Point and block preconditioners have been compared when the Laplace equation was solved. The block preconditioner preserves the properties of the point preconditioner. The most efficient preconditioner is an MILU factorization in which the last part is factorized exactly. This variant needs the least number of iterations and cpu-time and is almost grid independent. For the other variants of the block preconditioner grid refinement with a factor of two results in an increase in the number of iterations with a factor of two.

Applying the different variants of the preconditioner to the Navier-Stokes equations gives in some cases different results: for high Reynolds numbers the MILU factorization in which the last part is replaced by a diagonal matrix gives worse results than its ILU variant and for five levels in the ordering the MILU factorization leads to a stagnation of the convergence of the BiCGstab(2) method. The MILU factorization in which the last part is factorized exactly is still the most efficient preconditioner and shows almost grid independent behaviour for a small number of levels in the RRB ordering. For all

other variants of the block preconditioner grid refinement with a factor of two results in an increase in the number of iterations with a factor of about three.

The Boussinesq equations consist of four equations. Therefore the block size in the preconditioner is increased from three to four. The convergence behaviour remains the same. The block size has no influence on the efficiency of the preconditioner.

Several amendments can be made to improve the preconditioners used in this chapter. First, in the variants in which the last level is not factorized exactly the diagonal approximation can be changed to a more advanced one. Secondly, in the MILU variant the use of a relaxed modification can improve the convergence. Finally, we note that adding a time derivative to the system can make the approaches significantly more competitive.