Advances in Computational Mathematics 4(1995)27–49

Numerical solution of the incompressible Navier–Stokes equations by Krylov subspace and multigrid methods

S. Zeng, C. Vuik and P. Wesseling

Faculty of Technical Mathematics and Informatics, Delft University of Technology, Mekelweg 4, 2628 CD Delft, The Netherlands

We consider numerical solution methods for the incompressible Navier–Stokes equations discretized by a finite volume method on staggered grids in general coordinates. We use Krylov subspace and multigrid methods as well as their combinations. Numerical experiments are carried out on a scalar and a vector computer. Robustness and efficiency of these methods are studied. It appears that good methods result from suitable combinations of GCR and multigrid methods.

1. Introduction

We compare various iterative methods for linear systems resulting from discretization of the time-dependent incompressible Navier–Stokes equations. Before discretization the physical domain is mapped onto a computational domain consisting of a number of rectangular blocks. In this paper we restrict ourselves to the one-block case and two space dimensions. For the space discretization we use finite volumes and a staggered grid. For the time discretization we use the Euler Backward finite difference scheme together with pressure correction.

Krylov subspace and multigrid methods are two types of promising iterative methods for the solution of large unsymmetric non-diagonally dominant linear systems of algebraic equations. These types of methods are much used to solve discretized Navier–Stokes equations. Our research using Krylov subspace methods is described in [19–22] and using multigrid methods is described in [26–28,8–12]. Both types of method give satisfactory results. In this paper we compare the two approaches. Furthermore, we propose and compare combinations of these methods.

As Krylov subspace method we choose the GMRESR method [18] (a combination of GCR [2] and GMRES [15]). The reason for this is that GMRESR is more robust than the Bi-CGSTAB method [17] and requires less memory and CPU time than the GMRES method [15]. For the multigrid method we use a Galerkin coarse grid approximation and two different smoothers. Using the GMRESR method we observe that the number of iterations grows significantly

if the grid gets finer. For multigrid the number of iterations is in principle independent of the grid size. It appears that GMRESR is faster for medium grid sizes whereas multigrid (or combined) methods are faster for large grid sizes.

Since many of the faster computers are vector computers we also compare the vectorization properties of the different methods. Although we expect in the near future that parallel computers beat vector computers, the comparison will remain relevant because good vectorization properties imply in many cases good parallelization properties. Furthermore, vectorization aspects remain of interest because future high-performance parallel computing platforms will contain vector processors. Note that GMRESR is easy to vectorize, since most of its arithmetic operations are vector updates, vector–vector and matrix–vector operations. Vector length becomes large as the grid is refined, which improves speed on vector computers. With respect to multigrid we have the following choices:

- use a simple smoother, like point Jacobi, which is easily vectorized but not robust, or
- use a more complicated smoother, like ILU, which is robust but harder to vectorize.

A disadvantage of multigrid methods is that the occurrence of vectors of short length is inevitable, since use of coarse grids is necessary. This diminishes multigrid efficiency on vector computers.

The foregoing observations on the advantages and disadvantages of the two types of method suggest that combinations of them may be profitable. A combination of a Krylov subspace method with a multigrid method has already been described in [6]. In this paper the combined methods consist of an outer loop and an inner loop. The inner loop may be different in every outer iteration, so these combinations are very flexible. We investigate and compare the two types of methods and their combinations, specifically:

Method 1: GMRESR with ILU preconditioning;
Method 2: Multigrid with Jacobi line smoothing;
Method 3: Multigrid with ILU smoothing;
Method 4: GCR with Method 2 as inner loop;
Method 5: GCR with Method 3 as inner loop.

This paper is organized as follows. In section 2, the pressure correction scheme used and discretization are explained briefly. The five iterative methods are described in section 3, together with some analyses of their characteristics. In section 4 numerical results are presented, and analyses and discussions are given. Finally, we summarize observations and draw conclusions in section 5.

2. Equations and discretization

In this paper, general boundary-fitted coordinates are used to compute flows in complicated geometries. In general coordinates, the incompressible Navier–Stokes

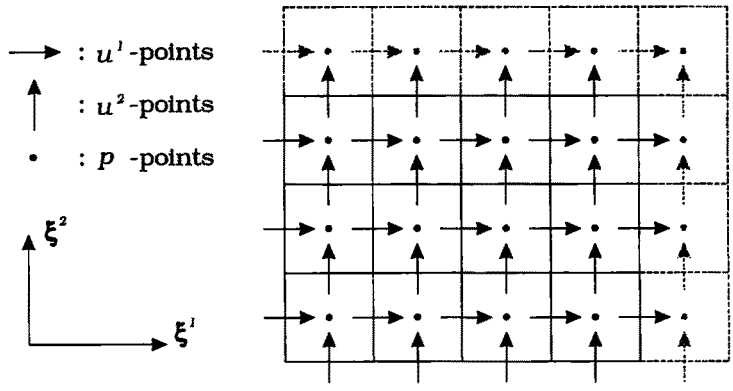

Figure 1. Staggered grid in computational domain. At certain boundaries, virtual cells are introduced, which are indicated by dashed lines.

equations are formulated in standard tensor notation as follows [16]:

momentum equations

$$\frac{\partial U^\alpha}{\partial t} + U^\beta U^\alpha_{,\beta} = -g^{\alpha\beta} p_{,\beta} + Re^{-1}(g^{\beta\gamma} U^\alpha_{,\beta} + g^{\alpha\beta} U^\gamma_{,\beta})_{,\gamma}, \tag{1}$$

continuity equation

$$U^\alpha_{,\alpha} = 0, \tag{2}$$

where U^α is the contravariant representation of the velocity vector field, p the pressure, Re the Reynolds number, and $g^{\alpha\beta}$ the metric tensor. The range of Greek indices is $\{1,2\}$, because we restricted ourselves to two space dimensions. As mentioned in section 1, a complicated physical domain is mapped onto a square computational domain. Figure 1 illustrates the computational domain and the staggered grid arrangement. We use a lexicographic ordering of the grid-points, where the points in ξ^1 direction are numbered first. Due to the use of virtual cells the number of u^1-, u^2- and p-points is the same. Using finite volume discretization in space and the backward Euler method for time discretization, we obtain the following discrete systems at each time step (see references [16,4,24] for details):

$$\frac{1}{\Delta t}\begin{pmatrix}\mathbf{u}^1\\ \mathbf{u}^2\end{pmatrix}^{n+1} - \frac{1}{\Delta t}\begin{pmatrix}\mathbf{u}^1\\ \mathbf{u}^2\end{pmatrix}^{n} = \begin{pmatrix}\mathbf{f}^1\\ \mathbf{f}^2\end{pmatrix}^{n+1} - \begin{pmatrix}\mathbf{A}^{11} & \mathbf{A}^{12} & \mathbf{A}^{13}\\ \mathbf{A}^{21} & \mathbf{A}^{22} & \mathbf{A}^{23}\end{pmatrix}\begin{pmatrix}\mathbf{u}^1\\ \mathbf{u}^2\\ \mathbf{p}\end{pmatrix}^{n+1}, \tag{3}$$

$$(\mathbf{A}^{31} \quad \mathbf{A}^{32})\begin{pmatrix}\mathbf{u}^1\\ \mathbf{u}^2\end{pmatrix}^{n+1} = 0, \tag{4}$$

where $\mathbf{u}^1$, $\mathbf{u}^2$ and $\mathbf{p}$ are algebraic vectors that approximate on the grid $\sqrt{g}U^1$ and $\sqrt{g}U^2$ and p, respectively, with $\sqrt{g}$ the Jacobian of the mapping, and $\mathbf{f}^1$ and $\mathbf{f}^2$ represent source terms. The nonlinear terms have been linearized with Newton's method. The linear operators $(\mathbf{A}^{31}\ \mathbf{A}^{32})$, resulting from discretization of the divergence operator in the continuity equation, and $\mathbf{A}^{13}$ and $\mathbf{A}^{23}$, resulting from

discretization of the gradients of the pressure in the momentum equations, do not depend on time. The remaining operators are time-dependent. The structure of stencils of the discrete operators are given by

$$[\mathbf{A}^{11}] = \begin{bmatrix} * & * & * \\ * & \underline{*} & * \\ * & * & * \end{bmatrix}, \quad [\mathbf{A}^{12}] = \begin{bmatrix} * & * & * & * \\ * & * & \underline{*} & * \end{bmatrix}, \quad [\mathbf{A}^{13}] = \begin{bmatrix} * & * \\ * & \underline{*} \\ * & * \end{bmatrix}, \tag{5}$$

$$[\mathbf{A}^{21}] = \begin{bmatrix} * & * \\ \underline{*} & * \\ * & * \\ * & * \end{bmatrix}, \quad [\mathbf{A}^{22}] = \begin{bmatrix} * & * & * \\ * & \underline{*} & * \\ * & * & * \end{bmatrix}, \quad [\mathbf{A}^{23}] = \begin{bmatrix} * & \underline{*} & * \\ * & * & * \end{bmatrix}, \tag{6}$$

$$[\mathbf{A}^{31}] = [\underline{*} \quad *], \quad [\mathbf{A}^{32}] = \begin{bmatrix} * \\ \underline{*} \end{bmatrix}, \tag{7}$$

where a star with an underscore corresponds to the center of the finite volume concerned.

Equations (3) and (4) are solved by the pressure correction method, as presented in [5], which consists of three steps. In the first step, the momentum equations are solved to give an intermediate value for the velocities, using the old pressure:

$$\begin{pmatrix} \frac{1}{\Delta t}\mathbf{I} + \mathbf{A}^{11} & \mathbf{A}^{12} \\ \mathbf{A}^{21} & \frac{1}{\Delta t}\mathbf{I} + \mathbf{A}^{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \end{pmatrix}^* = \begin{pmatrix} \mathbf{f}^1 \\ \mathbf{f}^2 \end{pmatrix}^{n+1} + \frac{1}{\Delta t} \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \end{pmatrix}^n - \begin{pmatrix} \mathbf{A}^{13} \\ \mathbf{A}^{23} \end{pmatrix} \mathbf{p}^n. \tag{8}$$

This equation system behaves like a discretization of a convection-diffusion equation. The main diagonal is enhanced by a contribution $1/\Delta t$ due to the time-derivative. Then the pressure equation, which is derived from the momentum equations (3) and the continuity equation (4), is solved to give the difference $\mathbf{p}^{n+1} - \mathbf{p}^n$:

$$(\mathbf{A}^{31} \quad \mathbf{A}^{32}) \begin{pmatrix} \mathbf{A}^{13} \\ \mathbf{A}^{23} \end{pmatrix} (\mathbf{p}^{n+1} - \mathbf{p}^n) = -\frac{1}{\Delta t} (\mathbf{A}^{31} \quad \mathbf{A}^{32}) \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \end{pmatrix}^*. \tag{9}$$

The coefficient matrix of $\mathbf{p}^{n+1} - \mathbf{p}^n$ does not change with time, and resembles a discretization of the Laplacian operator (in general coordinates), but is not symmetric. Finally, the velocities at time step $n+1$ are computed by means of

$$\begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \end{pmatrix}^{n+1} = \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \end{pmatrix}^* + \Delta t \begin{pmatrix} \mathbf{A}^{13} \\ \mathbf{A}^{23} \end{pmatrix} (\mathbf{p}^{n+1} - \mathbf{p}^n). \tag{10}$$

In the next section we describe the iterative methods used for the solution of (8) and (9).

3. Solution methods

In this section the iterative methods to be tested are described. The GMRESR

method combined with ILU type preconditioners is given in subsection 3.1. This is a summary of the methods described in [22]. In subsections 3.2.1 and 3.2.2, the multigrid methods using an alternating Jacobi line smoothing and an ILU smoothing are presented. New methods, consisting of combinations of GMRESR and multigrid are proposed in subsection 3.2.3. Finally, the vectorization properties of the iterative methods are summarized in subsection 3.3.

3.1. *Method 1: GMRESR with ILU preconditioning*

In section 2 we have seen that there are two types of linear systems to be solved: the momentum equations and the pressure equation. Each has its own characteristic properties. We use GMRESR for both but with different preconditioners. The GMRESR method is defined in [18], successfully applied to the Navier–Stokes equations in [20], and analyzed further in [19,21]. The GMRESR algorithm can be formulated as follows:

Algorithm GMRESR

$\mathbf{r}_0 = \mathbf{b} - \mathbf{A}\mathbf{x}_0, k = -1$
while $\|\mathbf{r}_{k+1}\|/\|\mathbf{r}_0\| > tol$ **do**
  $k := k + 1$
  apply one iteration of GMRES(m) to $\mathbf{A}\mathbf{y}_k = \mathbf{r}_k$ and
  denote the result by $\mathbf{u}_k^{(0)}$
  $\mathbf{c}_k^{(0)} = \mathbf{A}\mathbf{u}_k^{(0)}$
  for $i = 0, 1, \ldots, k-1$ **do**
    $\alpha_i = \mathbf{c}_i^T \mathbf{c}_k^{(i)}$
    $\mathbf{c}_k^{(i+1)} = \mathbf{c}_k^{(i)} - \alpha_i \mathbf{c}_i$; $\mathbf{u}_k^{(i+1)} = \mathbf{u}_k^{(i)} - \alpha_i \mathbf{u}_i$
  od
  $\mathbf{c}_k = \mathbf{c}_k^{(k)}/\|\mathbf{c}_k^{(k)}\|_2$; $\mathbf{u}_k = \mathbf{u}_k^{(k)}/\|\mathbf{c}_k^{(k)}\|_2$
  $\mathbf{x}_{k+1} = \mathbf{x}_k + \mathbf{u}_k \mathbf{c}_k^T \mathbf{r}_k$; $\mathbf{r}_{k+1} = \mathbf{r}_k - \mathbf{c}_k \mathbf{c}_k^T \mathbf{r}_k$
end while

GMRESR consists of a GCR outer loop and a GMRES inner loop. In this paper the GMRESR algorithm is used with the "*min alfa*" truncation strategy (see [19]). A truncation strategy is necessary to restrict the required memory. Truncation means the following: choose the number (*ntrunc*) of search directions ($\mathbf{u}_k$) that may be kept in memory. If the number of iterations becomes larger than *ntrunc*, a search direction $\mathbf{u}_j$ and its companion $\mathbf{c}_j$ ($=\mathbf{A}\mathbf{u}_j$) are overwritten by the new search direction $\mathbf{u}_{k+1}$ and $\mathbf{c}_{k+1}$. The *min alfa* truncation strategy is a method to decide which search direction should be discarded by the following criterion: find j such that $\alpha_j = \mathbf{c}_j^T \mathbf{c}_{k+1}^{(j)}$ satisfies the following equation:

$$|\alpha_j| = \min_{0 \leq i \leq ntrunc} |\alpha_i|. \tag{11}$$

To obtain an efficient solver, GMRESR is combined with a preconditioner. For the pressure equation we use the classical incomplete LU decomposition (all fill-in

is neglected). For the details of this preconditioner and the combination with GMRESR we refer to [22]. The preconditioning matrix must be stored and requires nine extra memory vectors. This memory is available, because the memory required to store the momentum matrix can be overwritten.

To save memory, we restrict ourselves to ILUD preconditionings for the momentum equations. In this type of preconditioning the off-diagonal parts of L and U are the same as that of the given matrix and only the diagonal is adapted. Thus only one extra memory vector is required. For further details we refer again to [22]. In the present paper, we use the RILUD_2 version for the momentum equations. One needs $2 \times ntrunc + m$ vectors in memory for the GMRESR(m) method for both systems of equations. In all the numerical experiments given in section 4, we use the GMRESR(5) method (so $m = 5$).

3.2. *Multigrid methods*

In this paper, we use multigrid methods consisting of the F-cycle with one pre- and one post-smoothing. In subsection 3.2.1 the coarse grid operators are defined. The two used smoothing operators are given in subsection 3.2.2, which lead to Methods 2 and 3.

3.2.1. Formulation of coarse grid operators

Coarse grid operators are formulated by means of Galerkin coarse grid approximation [23].

For brevity, we write equations (8) and (9) as

$$\begin{pmatrix} \mathbf{A}^{11} & \mathbf{A}^{12} \\ \mathbf{A}^{21} & \mathbf{A}^{22} \end{pmatrix} \begin{pmatrix} \mathbf{u}^1 \\ \mathbf{u}^2 \end{pmatrix} = \begin{pmatrix} \mathbf{f}^1 \\ \mathbf{f}^2 \end{pmatrix}, \tag{12}$$

$$\mathbf{A}^{33}\mathbf{p} = \mathbf{f}^3. \tag{13}$$

Let l be the grid index, with $l = 1$ indicating the coarsest grid. Galerkin coarse grid approximation is carried out from grid $l+1$ to grid l as follows:

momentum equations

$$\begin{pmatrix} \mathbf{A}^{11(l)} & \mathbf{A}^{12(l)} \\ \mathbf{A}^{21(l)} & \mathbf{A}^{22(l)} \end{pmatrix} = \begin{pmatrix} \mathbf{R}^1\mathbf{A}^{11(l+1)}\mathbf{P}^1 & \mathbf{R}^1\mathbf{A}^{12(l+1)}\mathbf{P}^2 \\ \mathbf{R}^2\mathbf{A}^{21(l+1)}\mathbf{P}^1 & \mathbf{R}^2\mathbf{A}^{22(l+1)}\mathbf{P}^2 \end{pmatrix},$$

$$\begin{pmatrix} \mathbf{f}^{1(l)} \\ \mathbf{f}^{2(l)} \end{pmatrix} = \begin{pmatrix} \mathbf{R}^1\mathbf{r}^{1(l+1)} \\ \mathbf{R}^2\mathbf{r}^{2(l+1)} \end{pmatrix} \tag{14}$$

and pressure equation

$$\mathbf{A}^{33(l)} = \mathbf{R}^3\mathbf{A}^{33(l+1)}\mathbf{P}^3, \quad \mathbf{f}^{3(l)} = \mathbf{R}^3\mathbf{r}^{3(l+1)}. \tag{15}$$

The $\mathbf{r}$'s are the residuals, for example, $\mathbf{r}^3 = \mathbf{f}^3 - \mathbf{A}^{33}\mathbf{p}$. Here, the $\mathbf{R}$'s and $\mathbf{P}$'s are restriction operators and prolongation operators, which are described below.

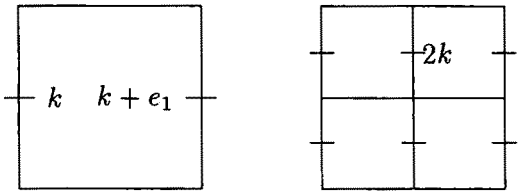

Figure 2. Cell-centered coarsening and correspondence between grid points (here for the $\mathbf{u}^1$ velocity unknowns) on a fine and a coarser grid; $e_1 = (1,0)$.

For an explanation of the principle of Galerkin coarse grid approximation, see [23]. In [28], an algorithm for efficient implementation of Galerkin coarse grid approximation is presented, and the properties of coarse grid operators are discussed (see also [25]) for various choices of restriction and prolongation operators.

Standard cell-centered coarsening is used: a cell on the next coarse grid is formed by taking the union of four fine grid cells, as illustrated in figure 2. The restriction operators $\mathbf{R}^1$ and $\mathbf{R}^2$ are for the momentum equations and $\mathbf{R}^3$ for the pressure equation. The prolongation operators $\mathbf{P}^1$, $\mathbf{P}^2$ and $\mathbf{P}^3$ are applied to $\mathbf{u}^1$, $\mathbf{u}^2$ and $\mathbf{p}$, respectively. The prolongation used for the coarse grid corrections is the same as in Galerkin coarse grid approximation.

The operators $\mathbf{R}^1$ and $\mathbf{R}^2$ use so-called hybrid interpolation, which, for example for $\mathbf{R}^1$, is obtained by using the adjoint of linear interpolation for $\mathbf{u}^1$ in direction 1 but the adjoint of piecewise constant interpolation in direction 2. Operator $\mathbf{R}^3$ is simply the adjoint of piecewise constant interpolation. Operators $\mathbf{R}^1$ and $\mathbf{R}^3$ are given by

$$[\mathbf{R}^1] = \frac{1}{2}\begin{bmatrix} we & 2 & \underline{we} \\ we & 2 & we \end{bmatrix}, \quad [\mathbf{R}^3] = \frac{1}{2}\begin{bmatrix} 1 & \underline{1} \\ 1 & 1 \end{bmatrix}, \tag{16}$$

where $w = 0$ when the "west" points are on or outside of the "west" boundary and $w = 1$ elsewhere, and similarly for s, e and n. $\mathbf{R}^2$ is similar to $\mathbf{R}^1$. The elements with an underscore correspond to the fine grid point $2k$ when restriction results in a function value in the coarse grid point k. The prolongation operators $\mathbf{P}^1$, $\mathbf{P}^2$ and $\mathbf{P}^3$ employ bilinear interpolation. The adjoints $\mathbf{P}^{1*}$ and $\mathbf{P}^{3*}$ of $\mathbf{P}^1$ and $\mathbf{P}^3$ are given by:

$$[\mathbf{P}^{1*}] = \frac{1}{8}\begin{bmatrix} nw & 2n & ne \\ (4-n)w & 2(4-n) & \underline{(4-n)e} \\ (4-s)w & 2(4-s) & (4-s)e \\ sw & 2s & se \end{bmatrix},$$

$$[\mathbf{P}^{3*}] = \frac{1}{16}\begin{bmatrix} nw & n(4-w) & n(4-e) & ne \\ (4-n)w & 16-4(n+w)+nw & \underline{16-4(n+e)+ne} & (4-n)e \\ (4-s)w & 16-4(s+w)+sw & 16-4(s+e)+se & (4-s)e \\ sw & s(4-w) & s(4-e) & se \end{bmatrix}, \tag{17}$$

and $\mathbf{P}^{2*}$ is similar to $\mathbf{P}^{1*}$. For a more detailed exposition of these transfer operators, see [23] and [28].

With our choice of $\mathbf{R}$'s and $\mathbf{P}$'s, the structures of all $\mathbf{A}$'s on coarse grids remain unchanged. The total number of coarse grid cells is about 1/3 of the number of cells on the finest grid. Therefore, to store the coarse grid operators for the momentum equations requires $17 \times 2/3 \cong 11$ vectors. The storage of the coarse grid operators for the pressure equation overwrites the storage of the momentum equations. This implies that at every time step, the coarse grid operators for the pressure equation are recomputed, which takes only little CPU time.

3.2.2. The smoothing operators

In this subsection we describe the smoothers which are used in the multigrid method: Jacobi smoothing and ILU smoothing. The reason for this choice is that Jacobi smoothing has good vectorization (parallelization) properties but is not robust, whereas the ILU smoothing is robust but not easily vectorized.

Method 2: Multigrid with Jacobi smoothing

Our Jacobi smoothing method consists of one horizontal Jacobi line iteration followed by one vertical Jacobi line iteration. The momentum equations are smoothed in a decoupled way, i.e., the two momentum equations are smoothed successively. In a horizontal smoothing iteration, mutually independent tri-diagonal systems have to be solved: $\mathbf{M}_j \delta \mathbf{x}_j = \mathbf{r}_j$ for a horizontal line j. The three non-zero elements at row i in $\mathbf{M}_j$ are denoted by $l_{i,j}$, $\hat{d}_{i,j}$, and $u_{i,j}$. The matrix $\mathbf{M}_j$ is factorized into:

$$\mathbf{M}_j = (\mathbf{L}_j + \mathbf{D}_j)\mathbf{D}_j^{-1}(\mathbf{D}_j + \mathbf{U}_j), \tag{18}$$

where $\mathbf{L}_j$ and $\mathbf{U}_j$ have only one non-zero diagonal below and above the main diagonal, equal to $l_{i,j}$ and $u_{i,j}$ and $\mathbf{D}_j$ is a diagonal matrix. Comparable formulas are used in a vertical smoothing iteration. Variables are updated after each horizontal and after each vertical step with a fixed underrelaxation factor $\omega = 0.7$.

Now we discuss the storage required by this smoother. $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ can overwrite $\mathbf{M}$, and also $\mathbf{r}$ and $\delta\mathbf{x}$ can share the same memory. So apart from the storage for the original system on all grids, additional memory is required to store $\mathbf{M}$ and $\mathbf{r}$. On the finest grid, the horizontal smoothing needs six vectors to store $\mathbf{M}$ for the momentum equations. For the coarser grids about 1/3 of the storage on the finest grid is required. Therefore, one needs $6 \times 4/3 = 8$ vectors to hold $\mathbf{M}$ for the horizontal smoothing. The same is true for the vertical smoothing. To store $\mathbf{r}$ one needs one vector on the finest grid, since the momentum equations are smoothed in a decoupled way. No additional storage for $\mathbf{r}$ on coarse grids is needed, as $\mathbf{r}$ (and also $\delta\mathbf{x}$) is used only once in a smoothing and can be overwritten after the smoothing. The solution of the pressure equation just uses the memory for the momentum equations. Therefore, this smoother needs additional storage of 17 vectors.

Method 3: Multigrid with ILU smoothing
Suppose that the equation to be smoothed is denoted by

$$\mathbf{A}\mathbf{x} = \mathbf{b}. \tag{19}$$

A smoothing iteration is given by

$$\delta\mathbf{x} = \mathbf{M}^{-1}(\mathbf{b} - \mathbf{A}\mathbf{x}), \quad \mathbf{x} := \mathbf{x} + \omega\delta\mathbf{x} \tag{20}$$

with $\omega = 0.8$ fixed. For the ILU smoothing we choose $\mathbf{M} = (\mathbf{L} + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U})$, where $\mathbf{L}$ and $\mathbf{U}$ are strictly lower and upper triangular matrices, and $\mathbf{D}$ a diagonal matrix. Matrices $\mathbf{L}$ and $\mathbf{U}$ have non-zero entries in the positions corresponding to the standard 9-point stencil pattern and are chosen such that the elements of $\mathbf{M}$ belonging to the 9-point pattern are equal to the corresponding elements of $\mathbf{A}$. The momentum equations are smoothed in the same decoupled manner as in Method 2. Again, factorization takes place only at the beginning of multigrid iterations for a time step, and $\mathbf{L}$, $\mathbf{D}$ and $\mathbf{U}$ are kept until the next time step. Here we need nine diagonals to store $\mathbf{M}$ for a momentum equation. Therefore, the additional memory required by this smoother is $9 \times 2 \times 4/3 + 1 = 25$ vectors.

3.2.3. The combined methods

The Jacobi line smoother is easy to vectorize but is less robust than ILU. Therefore it seems a good idea to combine line-Jacobi multigrid with GCR acceleration to obtain a fast and robust method. In the literature (other) combinations of Krylov subspace methods with multigrid are given in [6,7,13,14].

The methods presented below are very flexible. In many other combinations of Krylov subspace and multigrid methods, the inner loop procedures must be the same for every outer loop iteration. In these methods this is not necessary, so in different outer iterations one may use different inner loops, for instance a mix of GMRES and multigrid, or a different number of iterations with multigrid or multigrid with different smoothers, etc. The methods are based on the GMRESR idea where we use a GCR outer loop and a GMRES inner loop. The algorithms for the new methods are given below and only differ in the construction of the new search directions.

Method 4: GCR with Method 2 as inner loop
This method is obtained by replacing the statement

apply one iteration of GMRES(m) to $\mathbf{A}\mathbf{y}_k = \mathbf{r}_k$ and denote the result by $\mathbf{u}_k^{(0)}$

in Method 1 by

apply one iteration of Method 2 to $\mathbf{A}\mathbf{y}_k = \mathbf{r}_k$ and denote the result by $\mathbf{u}_k^{(0)}$.

The same truncation strategy as for GMRESR (see [19]) can be used. In our experiments, the number of iterations required for convergence was so small that no truncation was necessary. The memory requirements are $2 \times ntrunc$ vectors for the

Table 1
Megaflop rate of the vectorized preconditioner.

Grid	16 × 64	32 × 128	64 × 256	128 × 512
Mflop/s	15	22	32	35

GCR outer loop and 28 vectors for the multigrid inner loop (11 for the coarse grid operators + 17 for the smoother).

Method 5: GCR with Method 3 as inner loop
This method is obtained by replacing the statement

apply one iteration of Method 2 to $\mathbf{A}\mathbf{y}_k = \mathbf{r}_k$ and denote the result by $\mathbf{u}_k^{(0)}$

in Method 4 by

apply one iteration of Method 3 to $\mathbf{A}\mathbf{y}_k = \mathbf{r}_k$ and denote the result by $\mathbf{u}_k^{(0)}$.

The amount of memory required differs from that in Method 4, since Method 2 and Method 3 need different amounts of memory. The GCR outer loop requires still $2 \times ntrunc$ vectors, but the multigrid inner loop now needs 36 vectors (11 for the coarse grid operators + 25 for the smoother).

3.3. *Vectorization properties*

In this subsection we discuss the vectorization properties of the iterative methods given in subsection 3.1 and 3.2.

Method 1: GMRESR with ILU preconditioning
Most parts of the GMRESR method with ILU type preconditioning are well vectorizable. The only part which is hard to vectorize is the multiplication by the preconditioning matrix. This multiplication involves the solution of two linear systems, one with an upper and the other with a lower triangular matrix. In a straightforward algorithm recurrences prohibit vectorization. In [22] two ways are given to vectorize the solution of a triangular system: partial vectorization and a diagonal ordering of the computation. In this paper we use the second approach. Without vectorization the multiplication by the preconditioner has a speed of 9 Mflop/s on one processor of the Convex 3840. The speed of a vector update (which is fully vectorizable) is 35 Mflop/s. In table 1, the computing speeds are given for the vectorized version. For large gridsizes the speed of the preconditioner is equal to that of a vector update.

Method 2: Multigrid and Jacobi line smoothing
The computations of the coarse grid operators in (14) and (15) can be vectorized. With respect to the Jacobi line smoother, we note that for every horizontal line j a tridiagonal system $M_j \delta x_j = r_j$ have to be solved. The factorization of M_j as given in (18) can be computed by

$$d_{i,j} = \hat{d}_{i,j} - l_{i,j} u_{i-1,j} / d_{i-1,j}$$

Table 2
The CPU times and computational speeds in one alternating Jacobi line smoothing sweep on different grids.

	Process 1		Process 2	
Grid	CPU t	Mflop/s	CPU t	Mflop/s
16 × 64	0.020	7.8	0.0028	14.0
32 × 128	0.048	13.0	0.0078	20.0
64 × 256	0.13	18.0	0.024	25.0
128 × 512	0.56	17.0	0.094	25.0

for $i = 1, 2, \ldots, n_1$, where all terms with index outside the domain are deleted. We achieve vectorization (parallelization) by nesting the loops as follows:

for $i = 1, 2, \ldots, n_1$ do
  for $j = 1, 2, \ldots, n_2$ do
    $d_{i,j} = \hat{d}_{i,j} - l_{i,j} u_{i-1,j} / d_{i-1,j}$
  od
od.

Forward and backward substitution can be handled in a similar fashion. Note that in our case (Fortran, natural ordering of gridpoints), the memory is accessed with stride n_1 for the horizontal smoothing, but it is accessed contiguously for the vertical smoothing.

The most CPU time consuming parts of a smoothing sweep are: Process 1: computation of $\mathbf{r}$, and Process 2: the solution of the tridiagonal systems. Since we use an alternative direction line smoothing, a stride equal to n_1 cannot be avoided. We have arranged storage of $\mathbf{M}$ and $\mathbf{r}$ such that in Process 2 memory is accessed contiguously for both the horizontal and the vertical smoothings. This speeds up Process 2 but slows down Process 1 somewhat (this can be verified by comparing Process 1 with the corresponding process in the ILU smoothing in the next subsection). In table 2, we present for one smoothing sweep the CPU times in seconds spent on Process 1 and 2, and the corresponding Megaflop rate on the Convex C3840. Note that Process 1, which essentially consists of two matrix vector products, is the most time consuming part. For both parts the Mflop rate grows for increasing gridsize. Finally the Mflop rate of Process 2 shows that the solution of the tridiagonal system is well vectorized.

Method 3: Multigrid with ILU smoothing

Using an ILU smoothing the most CPU time consuming parts are: Process 1: computation of $\mathbf{r}$ and Process 2: the solution of $(\mathbf{L} + \mathbf{D})\mathbf{D}^{-1}(\mathbf{D} + \mathbf{U})/\delta\mathbf{x} = \mathbf{r}$. The corresponding CPU times and Mflop rates are given in table 3. Indeed the Mflop rate of Process 1 is somewhat better than that used in Method 2, but the Mflop rate of Process 2 is only 1/3 of that used in Method 2. In this smoothing, Process 2 is the time dominant part.

Table 3
The CPU times and computing speed for one ILU smoothing step on different grids.

	Process 1		Process 2	
Grid	CPU t	Mflop/s	CPU t	Mflop/s
16 × 64	0.010	8.5	0.024	2.5
32 × 128	0.024	14.0	0.052	4.4
64 × 256	0.072	18.0	0.13	6.8
128 × 512	0.24	21.0	0.41	8.7

For both methods, but especially for Process 2 of Method 3, we observe that the computing speed on coarse grids is not high due to smaller vector lengths. However coarse grids cannot be avoided in multigrid. Choosing the coarsest grid not too coarse and using methods faster than the smoother on the coarsest grid could improve efficiency on vector computers. As we will see, Method 1 could be a suitable coarsest grid solver.

Methods 4, 5: The combined methods
Since the overhead of the GCR outer loop is small and well vectorizable, the vectorization properties of these methods is mainly determined by the multigrid part. This implies that Method 4 is better vectorizable than Method 5.

4. Numerical experiments

4.1. *Test problems*

We consider four test problems, which are the square driven cavity problem with uniform and non-uniform grids, the oblique driven cavity problem and the L-shaped driven cavity problem, as illustrated in figure 3. We refer to these problems as Problems 1 to 4, respectively. These problems give rise to different difficulties. In the first two problems, the coordinates are orthogonal, and as a result there are no mixed derivatives. By adding more difficulty, the second problem has strongly stretched cells near boundaries, where the maximum cell aspect ratio is 100. The last two test problems both introduce mixed derivatives, the last one being typical of a general case.

We study these problems for $Re = 1, 1000$, $\Delta t = 0.0625, 0.125, 0.25$, and three grid sizes 32×32, 64×64, 128×128 (sometimes also 256×256). The coarsest grid is 2×2, on which exact solution takes place by using a direct solver. The number of time steps is fixed at 40. This number is a rather arbitrary choice, because our purpose here is not to solve problems until steady state, but to investigate the performance (efficiency and robustness) of solution methods. Based on numerical experiments, the following stop criterion is chosen: the iterative solution of the systems at each time step is terminated if the ratio of the norm $||\mathbf{r}||$ of the

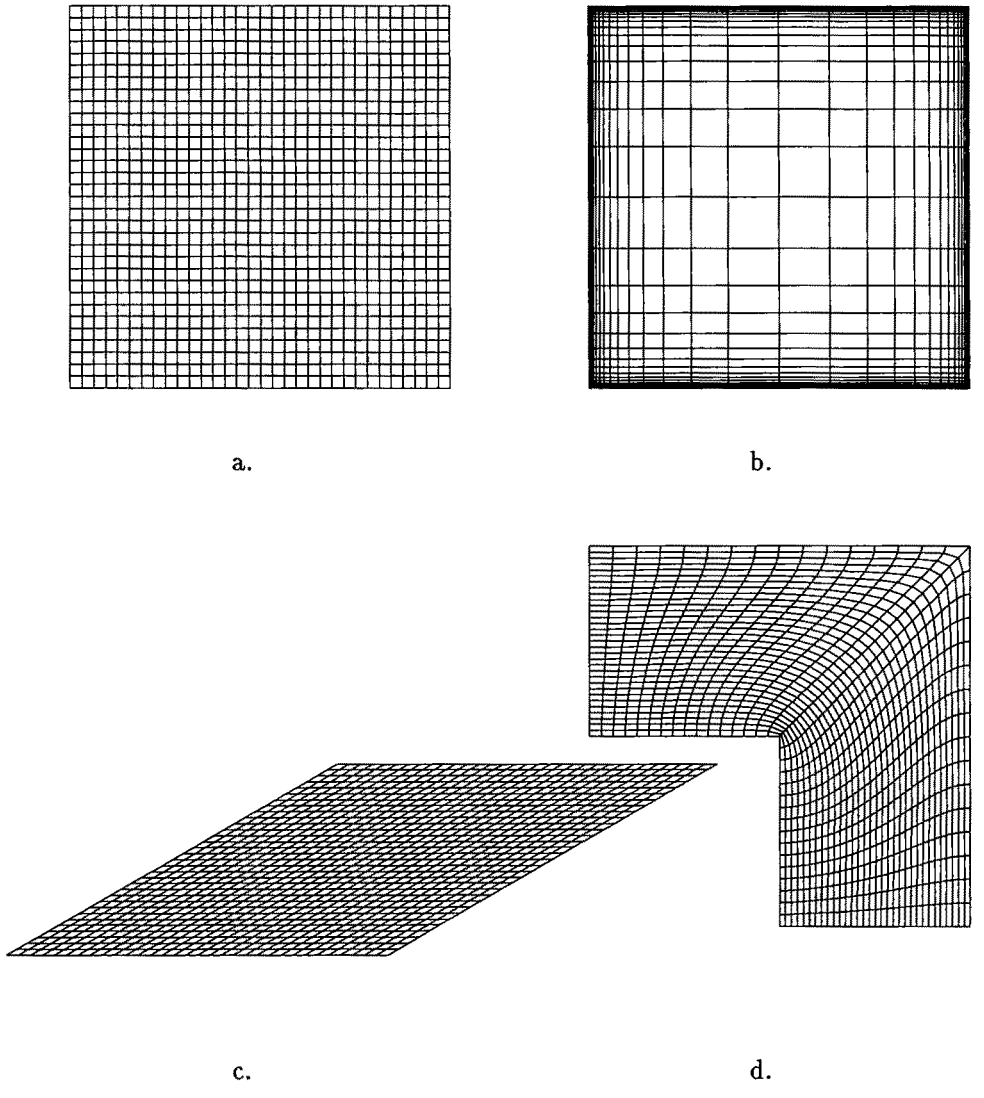

Figure 3. Grids of size 32×32 for the four test problems: (a) The square driven cavity problem; (b) The non-uniform square driven cavity problem; (c) The oblique driven cavity problem; (d) The L-shaped driven cavity problem.

residual to the norm $||\mathbf{r}_0||$ of the residual at the beginning of the present time step satisfies $||\mathbf{r}||/||\mathbf{r}_0|| < tol$, with $tol = 10^{-4}$ for the momentum equations and $tol = 10^{-6}$ for the pressure equation.

In subsection 4.2 experiments on a scalar computer are described whereas subsection 4.3 contains the results on a vector computer.

4.2. *Experiments on a scalar computer*

In this subsection we present numerical experiments on an HP 735 computer. We

Table 4
The results for Method 1 on a 128 × 128 grid and $Re = 1000$.

Δt	k_v	t_v
0.0625	2	97
0.125	4	132
0.25	6	213

have run all methods described in section 3 for the test problems given in subsection 4.1. For brevity, here we only present a representative subset of the results. In subsection 4.2.1 the momentum equations are considered, whereas in subsection 4.2.2 we show results for the pressure equation. Finally, subsection 4.2.3 contains results for an entering flow problem.

4.2.1. The momentum equations

The properties of the linear systems originating from the discretized momentum equations that influence the iterative solvers depend on: the size of the time step, the Reynolds number, the grid size, and the shape of the space domain. Below the influence of these parameters is considered in more detail. In the first parts we restrict ourselves to Problem 3, only in the final part results are given for all test problems. The reason for this is that the results for the other problems are comparable with those of Problem 3.

Dependence on Δt

As we have already said the main diagonal of the momentum matrix is enhanced by a contribution $1/\Delta t$ due to the time derivative. So for small Δt the matrix is diagonal dominant. In our examples Δt is chosen relatively small, since for large Δt the pressure correction scheme diverges. It appears that Methods 2 to 5 are more or less independent of the time step. Method 1 depends on the time step as can be seen from table 4. In this table t_v is the total CPU time for the solution of the momentum equation over 40 time steps and k_v is the number of iterations at the final step. Note that the number of iterations grows if Δt increases.

Dependence on the Reynolds number

Only Methods 1 and 2 show a clear dependence on the Reynolds number. For both methods the number of iterations is less for $Re = 1000$ than for $Re = 1$, see table 5.

Table 5
The results for Methods 1 and 2 on a 128 × 128 grid and $\Delta t = 0.0625$.

Reynolds number	Method 1		Method 2	
	k_v	t_v	k_v	t_v
1	14	774	6	741
1000	2	97	4	508

Table 6
The results for Method 1 using $\Delta t = 0.0625$ and $Re = 1$.

Grid size	k_v	t_v
32×32	5	7
64×64	8	74
128×128	14	774

Note that Method 1 converges very fast for $Re = 1000$. However we have to keep in mind that one outer iteration of GMRESR(5) consists of 5 iterations of GMRES. It appears that the eigenvalues of the preconditioned matrix are well clustered around 1, which explains the fast convergence of GMRESR(5). Up till now we are not able to explain why the eigenvalues are close to one.

Grid size dependence
Again, only Method 1 depends on the grid size whereas the other methods are grid size independent. For the low Reynolds number some results are given in table 6. As expected, the number of iterations increases for increasing grid size.

Problem dependence and comparison
For a comparison of the various methods on the four test problems we plot the CPU time on an HP735 per grid point for 40 time steps against the grid size. In these figures we use the following symbols:

Method 1: solid lines and point marks,
Method 2: dotted lines and circles,
Method 3: dashed lines and stars,
Method 4: dotted lines and plus marks,
Method 5: dashed lines and x-marks.

Where no symbols are shown they are off-scale. For $Re = 1$ the results are given in figure 4 and for $Re = 1000$ the results are given in figure 5.

First, we discuss the combination of GCR and multigrid. From figures 4 and 5 it appears that the GCR acceleration of the Jacobi smoothed multigrid is better than multigrid itself. If the smoother is sufficiently powerful, as for instance for Method 3 where we use an ILU smoother, then the combination of GCR and multigrid gives a slightly worse performance. In these cases the number of iterations is the same but the CPU time increases somewhat due to the GCR overhead.

Secondly, we compare Method 1 with the best multigrid method: Method 3. Note that for Method 3 the CPU time per grid point is more or less the same for every problem. For Method 1 there is more variation: the CPU time increases for a larger grid size and a smaller Reynolds number. For a large Reynolds number Method 1 is much faster than Method 3 (except Problem 2). For a small Reynolds number Method 1 is more efficient for medium grid sizes, whereas Method 3 is the best method for large grid sizes. For Problems 1, 2 and 3 the

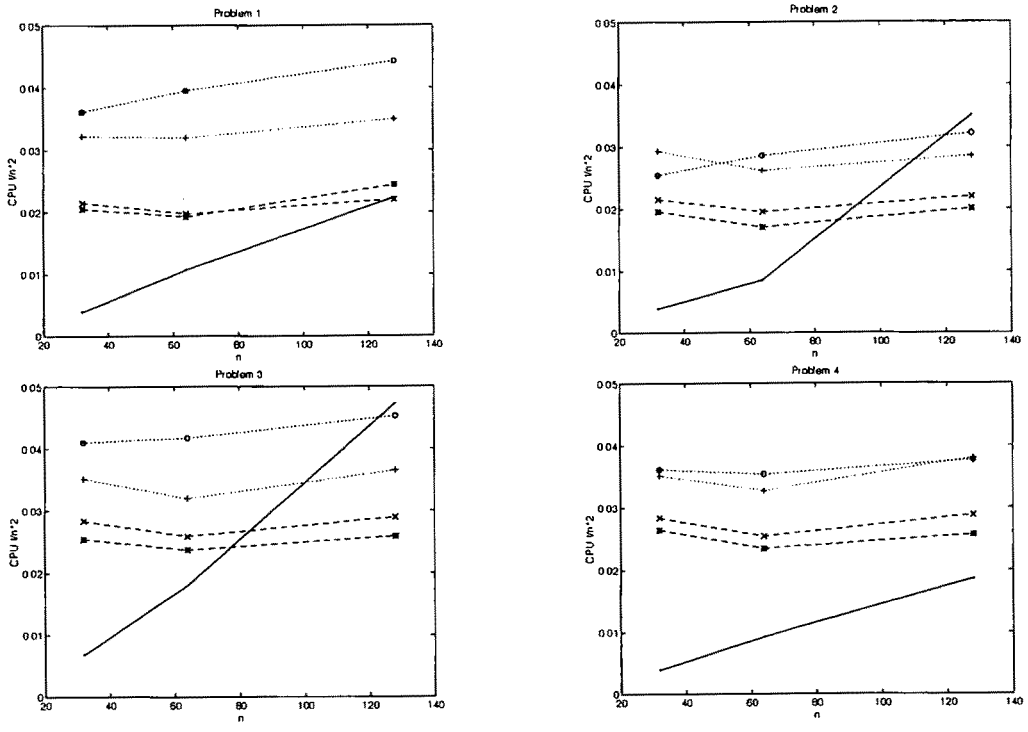

Figure 4. CPU times per grid point for the momentum equation during 40 time steps, for $Re = 1$ and $\Delta t = 0.0625$.

break-even point is in the range [64,128] and for Problem 4 the break-even point is in the range [128,256].

Finally, we discuss robustness. Methods 3 and 5 are equally robust, because for all test cases they both work well. Method 1 is a little less robust, since for Problem 2 it has several failure cases (not shown here) when Δt is large, on the fine grid for Re small or on the coarse grids for Re large. The least robust method is Method 2; it suffers from convergence problems when either the grid is refined or Δt is large for some problems. But when it is combined with GCR, resulting in Method 4, robustness is improved very much. Sometimes when Method 2 fails to work, Method 4 still works rather satisfactorily. However, Method 4 falls behind Methods 1, 3 and 5 for Re large.

4.2.2. The pressure equation

The properties of the discretized pressure equation depends only on: the grid size and the shape of the space domain.

Grid size dependence

The multigrid and combined methods require the same number of iterations for increasing grid size. Again, Method 1 depends on the grid size: the number

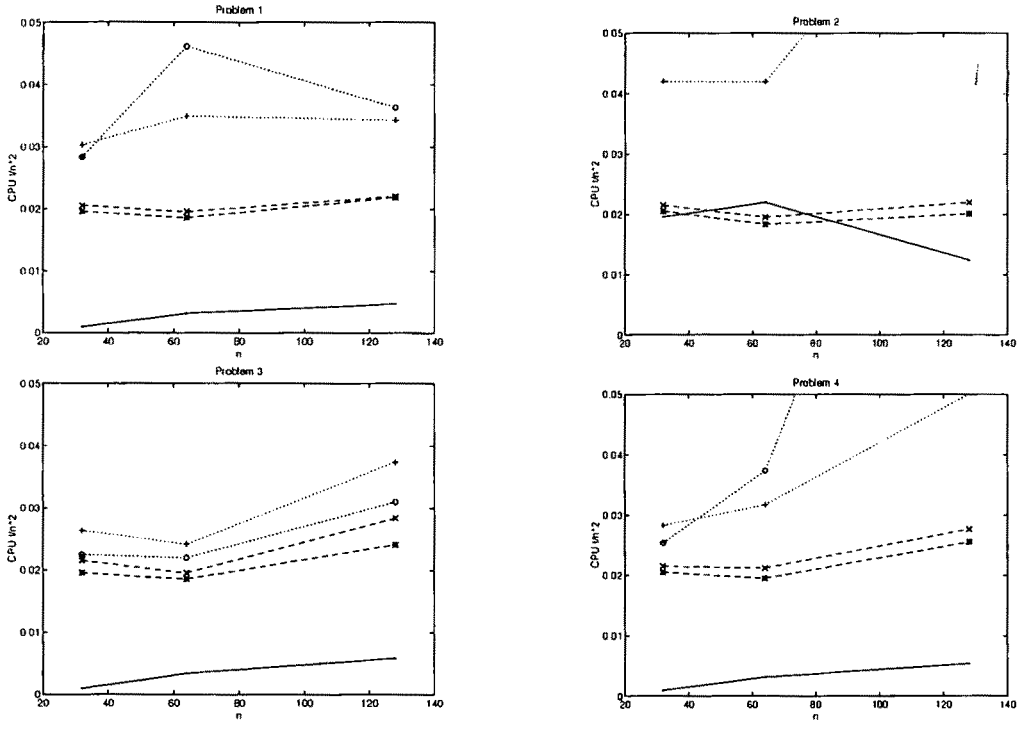

Figure 5. CPU times per grid point for the momentum equations during 40 time steps for $Re = 1000$ and $\Delta t = 0.0625$.

of iterations grows for increasing grid size. This is illustrated by table 7, where the results for Problem 3 are given. The total CPU time to solve the pressure equation is denoted by t_p and k_p is the number of iterations at the final step.

Problem dependence and comparison

The CPU time on an HP 735 per grid point for 40 time steps is shown in figure 6. It appears from figure 6 that for both smoothers the combination of GCR and multigrid is more efficient than multigrid itself. Especially in Problem 3, Method 4 is two times as fast as Method 2. Also for the strong ILU smoother the CPU time for Method 5 is considerably less than for Method 3. Note that the CPU time for Method 1 is approximately the same for Problem 2 to 4 but it is halved

Table 7
The results for Method 1 used in Problem 3.

Grid size	k_p	t_p
32 × 32	9	7
64 × 64	13	57
128 × 128	22	642

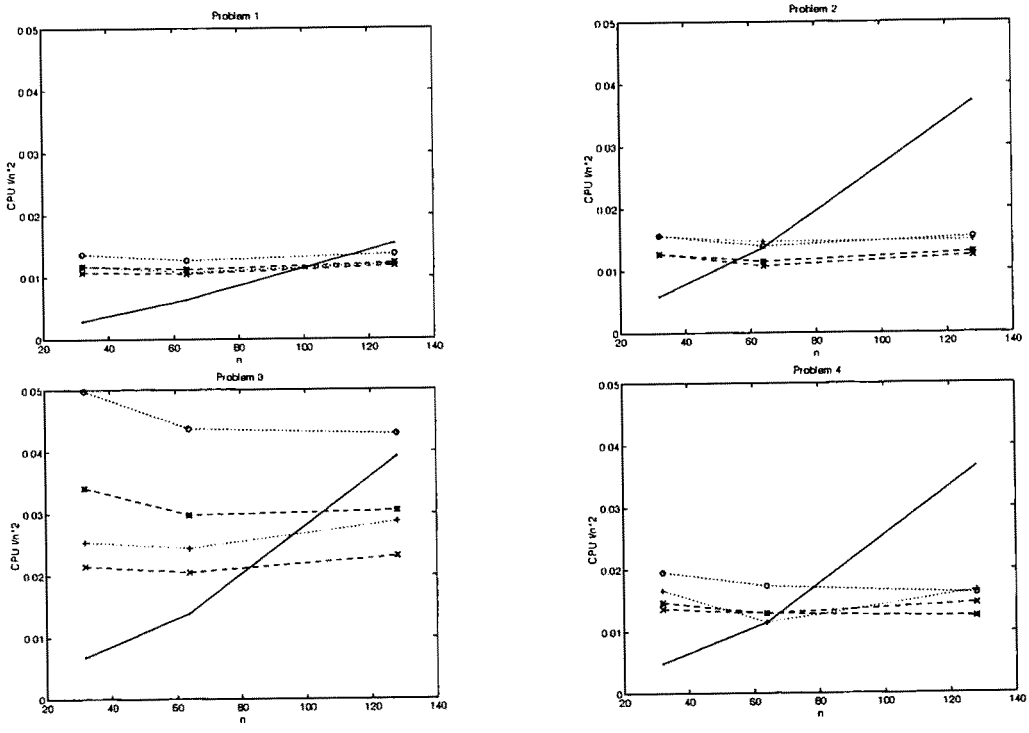

Figure 6. CPU times per grid point for the pressure equation during 40 time steps.

for Problem 1. With respect to the multigrid (or combined) method we observe that Problem 3 takes the most CPU time.

Finally, we compare Method 1 with the best multigrid method: Method 5. It appears that Method 1 is more efficient for medium grid sizes, whereas Method 5 is more efficient for large grid sizes. For Problems 2 and 4 the break-even points is in range [32, 64] whereas for Problems 1 and 3 the break-even point is in the range [64, 128]. For the pressure equation, Method 1 has a superlinear convergence behaviour [22], which means the reduction of residuals is faster in later iterations than in the first ones. Since the multigrid and combined method are linear convergent this implies that decreasing the termination criterion *tol* would benefit Method 1 and vice-versa.

4.2.3. A test problem with an entering flow

In subsections 4.2.1 and 4.2.2 we have considered examples of driven cavity flows. In other test problems we have observed comparable results. In order to illustrate this we present in this subsection results for an entering flow, namely a Backward Facing Step problem [3]. The 46×16 grid for the problem is given in figure 7. The results for the momentum and pressure equations are given in figure 8. The results are in good agreement with the observations made in subsections 4.2.1 and 4.2.2.

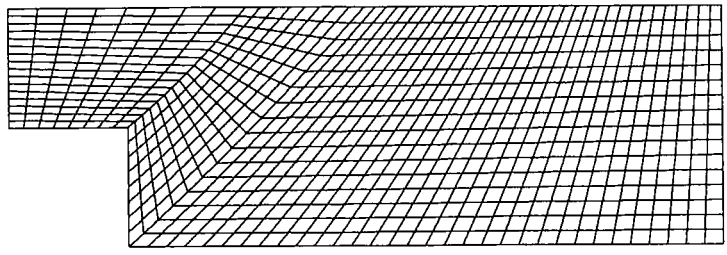

Figure 7. The 48 × 16 grid for the Backward Facing Step problem.

4.3. *Experiments on a vector machine*

In this subsection we report on some experiments on a Convex C3840. First we compare Methods 1, 3 and 5, because they are the best methods on the scalar machine and have different vectorization properties. Thereafter, Methods 3 and 5 are compared with Methods 2 and 4 to analyse the performance of methods using a weaker smoother but with greater vectorization potential and using a stronger smoother but with smaller vectorization capacity.

Comparing the best methods

In figures 9 and 10 we present the CPU time per grid point against grid size for Problems 3 and 4. To show the effect of an increasing vector length computations on a 256 × 256 grid are included for Problem 4 (note the difference in horizontal scale). From these figures it appears that the behaviour of the different methods is comparable to that on a scalar machine: the efficiency of Method 1 deteriorates and that of Methods 3 and 5 improves with grid refinement. Due to the good vectorization properties of the Krylov methods the break-even point moves to finer grids and the GCR overhead for the combined methods becomes negligible. Finally, the curves for Methods 3 and 5 become flatter when going to finer grids, which indicates that the efficiency gain from a larger vector length is gradually exhausted.

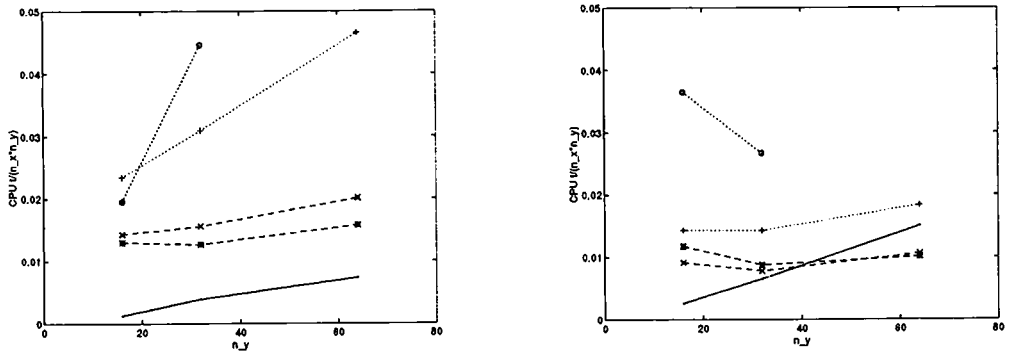

Figure 8. CPU times per grid point for the Backward Facing Step problem during 40 time steps, for $Re = 150$ and $\Delta t = 0.0625$. Left: the momentum equations, right: the pressure equation.

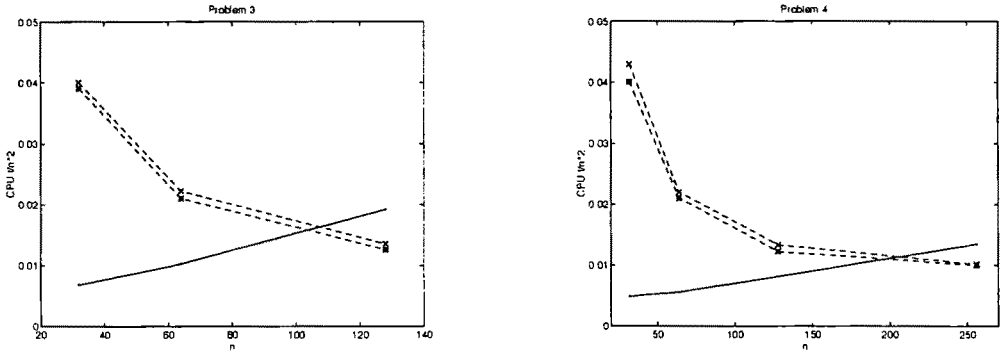

Figure 9. CPU times per grid point for the momentum equations during 40 time steps for $Re = 1$ and $\Delta t = 0.0625$.

Comparing the vectorization properties of the smoothers

It appears that the higher Mflop rate of Methods 2 and 4 does not compensate the slower rate of convergence, although on the vector machine they compete better than on the scalar machine. This is true for all test problems and is illustrated with the momentum equations of Problem 4 in figure 11. Note that for a low Reynolds number Methods 2 to 5 are comparable, but for a high Reynolds number Methods 3 and 5 are superior to Methods 2 and 4. Method 2 does not work on finer grids and even fails on the 256×256 grid.

5. Conclusions

We have investigated numerically five iterative methods, namely, Method 1: GMRESR: GCR with GMRES as inner loop, Method 2: multigrid with a Jacobi line smoothing, Method 3: multigrid with an ILU smoothing, Method 4: GCR with multigrid with Jacobi line smoothing as inner loop and Method 5: GCR with multigrid with ILU smoothing as inner loop, in the context of application to the solution of the incompressible Navier–Stokes equations in general

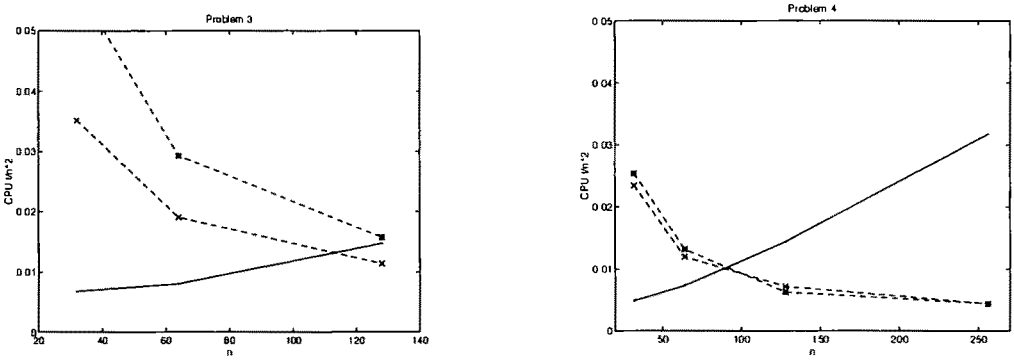

Figure 10. CPU times per grid point for the pressure equation during 40 time steps.

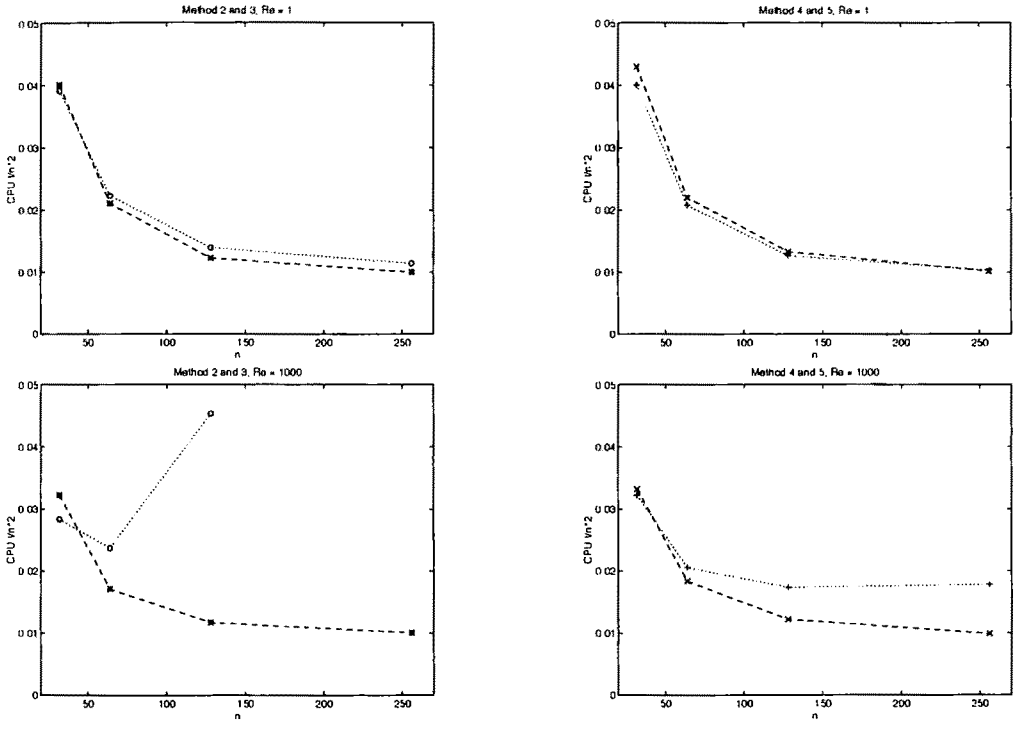

Figure 11. CPU times per grid point for the momentum equations during 40 time steps for Problem 4 and $\Delta t = 0.0625$. Above: $Re = 1$ below $Re = 1000$.

coordinates on staggered grids, using the pressure correction method in the time-dependent case.

From our numerical experiments we draw the following conclusions:

- For the solution of the momentum equations with a high Reynolds number Method 1 is the best method.
- Solving the momentum equations for a low Reynolds number Method 1 is faster for medium sized grids, whereas Method 3 is the best method for the large sized grids.
- For the pressure equation Method 1 is also optimal for medium grid sizes. For large grid sizes Method 5 is the most robust and efficient method.
- The GCR outerloop of Methods 4 and 5 speeds up the rate of convergence, especially for weak smoothers (Method 4).

Finally, we remark that the break-even point, where the efficiency of the Krylov subspace method is equal to that of the multigrid method depends on many factors. Some of them are: the domain of the test problem, the termination criterion, the Reynolds number, the computer used (scalar, vector, or parallel) etc. In section 4 we have investigated numerically in which direction the break-even point moves depending on the change of one of these factors.

References

[1] C.C. Ashcraft and R.G. Grimes, On vectorizing incomplete factorization and SSOR preconditioners, SIAM J. Sci. Stat. Comp. 9 (1988) 122–151.
[2] S.C. Eisenstat, H.C. Elman and M.H. Schultz, Variable iterative methods for non-symmetric systems of linear equations, SIAM J. Numer. Anal. 20 (1983) 345–357.
[3] K.J. Morgan, J. Periaux and F. Thomasset (eds.), *Analysis of Laminar Flow over a Backward Facing Step*, GAMM Workshop held at Bièvres (Fr.) (Vieweg, Braunschweig, 1984).
[4] A.E. Mynett, P. Wesseling, A. Segal and C.G.M. Kassels, The ISNaS incompressible Navier–Stokes solver: invariant discretization, Appl. Sci. Res. 48 (1991) 175–191.
[5] J.J.I.M. van Kan, A second-order accurate pressure-correction scheme for viscous incompressible flow, SIAM J. Sci. Stat. Comput. 7 (1986) 870–891.
[6] R. Kettler, Analysis and comparison of relaxation schemes in robust multigrid and conjugate gradient methods, in: *Multigrid Methods*, eds. W. Hackbusch and U. Trottenberg, Lecture Notes in Mathematics 960 (Springer, Berlin, 1982) pp. 502–534.
[7] R. Kettler, Linear multigrid methods for numerical reservoir stimulation, Ph.D. Thesis, Delft University of Technology (1987).
[8] C.W. Oosterlee and P. Wesseling, A multigrid method for an invariant formulation of the incompressible Navier–Stokes equations in general co-ordinates, Commun. Appl. Numer. Meth. 8 (1992) 721–734.
[9] C.W. Oosterlee and P. Wesseling, A robust multigrid method for a discretization of the incompressible Navier–Stokes equations in general coordinates, in: *Computational Fluid Dynamics*, eds. Ch. Hirsch, J. Périaux and W. Kordulla (Elsevier, Amsterdam, 1992) pp. 101–108.
[10] C.W. Oosterlee, Robust multigrid methods for the steady and unsteady incompressible Navier–Stokes equations in general coordinates, Ph.D. Thesis, Delft University of Technology (1993).
[11] C.W. Oosterlee and P. Wesseling, A robust multigrid method for a discretization of the incompressible Navier–Stokes equations in general coordinates, Impact. Comp. Sci. Eng. 5 (1993) 128–151.
[12] C.W. Oosterlee and P. Wesseling, Multigrid schemes for time-dependent incompressible Navier–Stokes equations, Impact. Comp. Sci. Eng. 5 (1993) 153–175.
[13] P. Sonneveld, P. Wesseling and P.M. de Zeeuw, Multigrid and conjugate gradient methods as convergence acceleration techniques, in: *Multigrid Methods for Integral and Differential Equations*, eds. D.J. Paddon and H. Holstein (Clarendon Press, Oxford, 1985) pp. 117–168.
[14] P. Sonneveld, P. Wesseling and P.M. de Zeeuw, Multigrid and conjugate gradient acceleration of basic iterative methods, in: *Numerical Methods for Fluid Dynamics II*, eds. K.W. Morton and M.J. Baines (Clarendon Press, Oxford, 1986) pp. 347–368.
[15] Y. Saad and M.H. Schultz, GMRES: a generalized minimal residual algorithm for solving non-symmetric linear systems, SIAM J. Sci. Stat. Comp. 7 (1986) 856–869.
[16] A. Segal, P. Wesseling, J. van Kan, C.W. Oosterlee and C.G.M. Kassels, Invariant discretization of the incompressible Navier–Stokes equations in boundary fitted co-ordinates, Int. J. Numer. Meth. Fluids 15 (1992) 411–426.
[17] H.A. van der Vorst, Bi-CGSTAB: a fast and smoothly converging variant of Bi-CG for solution of non-symmetric linear systems, SIAM J. Sci. Stat. Comp. 13 (1992) 631–644.
[18] H.A. van der Vorst and C. Vuik, GMRESR: A family of nested GMRES methods, Numer. Lin. Alg. Appl. 1 (1994) 369–386.
[19] C. Vuik, Further experiences with GMRESR, Supercomputer 55 (1993) 13–27.
[20] C. Vuik, Solution of the discretized incompressible Navier–Stokes equations with the GMRES method, Int. J. Numer. Meth. Fluids 16 (1993) 507–523.
[21] C. Vuik, New insights in GMRES-like methods with variable preconditioners, Report 93-10, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands (1993), to appear in J. Comp. Appl. Math.

[22] C. Vuik, Fast iterative solvers for the discretized incompressible Navier–Stokes equations, Report 93-98, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands (1993), to appear in Int. J. Numer. Meth. Fluids.
[23] P. Wesseling, *An Introduction to Multigrid Methods* (Wiley, Chichester, 1992).
[24] P. Wesseling, A. Segal, J. van Kan, C.W. Oosterlee and C.G.M. Kassels, Finite volume discretization of the incompressible Navier–Stokes equations in general coordinates on staggered grids, Comp. Fluid Dyn. J. 1 (1992) 27–33.
[25] P.M. De Zeeuw, Matrix-dependent prolongations and restrictions in a block multigrid method solver, J. Comp. Appl. Math. 3 (1990) 1–7.
[26] S. Zeng and P. Wesseling, An ILU smoother for the incompressible Navier–Stokes equations in general coordinates, Int. J. Numer. Meth. Fluids 20 (1995) 59–74.
[27] S. Zeng and P. Wesseling, Numerical study of a multigrid method with four smoothing methods for the incompressible Navier–Stokes equations in general coordinates, in: *6th Copper Mountain Conf. on Multigrid Methods*, eds. N. Duane Melson, T.A. Manteuffel and S.F. McCormick, NASA Conference Pub. 3224 (1993) pp. 691–708.
[28] S. Zeng and P. Wesseling, Multigrid solution of the incompressible Navier–Stokes equations in general coordinates, SIAM J. Num. Anal. 31 (1994) 1764–1784.
[29] S. Zeng, C. Vuik and P. Wesseling, Solution of the incompressible Navier–Stokes equations in general coordinates by Krylov subspace and multigrid methods, Report 93-64, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands (1993).
[30] S. Zeng, C. Vuik and P. Wesseling, Further investigation on the solution of the incompressible Navier–Stokes equations by Krylov subspace and multigrid methods, Report 93-93, Faculty of Technical Mathematics and Informatics, TU Delft, The Netherlands (1993).