INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2000; **33**: 453–473

Numerical performance of smoothers in coupled multigrid methods for the parallel solution of the incompressible Navier–Stokes equations

Volker John*,[1] and Lutz Tobiska

Institut für Analysis und Numerik, Otto-von-Guericke-Universität Magdeburg, Postfach 4120, 39016 Magdeburg, Germany

SUMMARY

In recent benchmark computations [Schäfer M, Turek S. The benchmark problem 'Flow around a cylinder'. In *Flow Simulation with High-Performance Computers II*, Hirschel EH (ed.), vol. 52 of Notes on Numerical Fluid Mechanics. Vieweg: Wiesbaden, 1996; 547–566], coupled multigrid methods have been proven as efficient solvers for the incompressible Navier–Stokes equations. A numerical study of two classes of smoothers in the framework of coupled multigrid methods is presented. The class of Vanka-type smoothers is characterized by the solution of small local linear systems of equations in a Gauss–Seidel manner in each smoothing step, whereas the Brass–Sarazin-type smoothers solve a large global saddle point problem. The behaviour of these smoothers with respect to computing times and parallel overheads is studied for two-dimensional steady state and time-dependent Navier–Stokes equations. Copyright © 2000 John Wiley & Sons, Ltd.

KEY WORDS: Braess–Sarazin-type smoothers; incompressible Navier–Stokes equations; parallel coupled multigrid methods; Vanka-type smoothers

1. INTRODUCTION

Over the last decades, various methods for the numerical solution of the incompressible Navier–Stokes equations have been developed. Solvers and discretizations of the imcompressible Navier–Stokes equations were already compared by benchmark computations, e.g. within the priority research program 'Flow simulation with high-performance computers' of the Deutsche Forschungsgemeinschaft (DFG). As one result of the critical evaluation of these benchmark computations [1], coupled multigrid methods seem to be among the best classes of solvers that are known currently.

* Correspondence to: Institut für Analysis und Numerik, Otto-von-Guericke-Universität Magdeburg, Postfach 4120, 39016 Magdeburg, Germany. Tel.: +49 391 6712633; fax: +49 391 6718073.
[1] E-mail: john@mathematik.uni-magdeburg.de

Received November 1998
Revised March 1999

The efficiency of a multigrid method is essentially influenced by the smoothing algorithm. Two classes of smoothers for coupled multigrid methods are studied numerically in this paper; on the one hand, smoothers of Vanka type [2], and on the other hand, Braess–Sarazin-type smoothers [3]. The Vanka-type smoothers can be considered as block Gauss–Seidel methods, where in each smoothing step a number of small linear systems of equations have to be solved. In contrast, Braess–Sarazin-type smoothers solve a large saddle point problem in each smoothing step. This saddle point problem is easier to solve than the saddle point problem arising in the discretization of the linearized Navier–Stokes equations and it will be solved by a pressure Schur complement method.

Vanka-type smoothers in coupled multigrid methods for solving the Navier–Stokes equations have been studied in a number of papers, e.g. References [4–7]. Pressure Schur complement schemes as smoothers in coupled multigrid methods have been reported on as much in the literature. Gjesdal and Lossius [8] studied several SIMPLE-type methods and mentioned that they may not be as efficient as Vanka-type smoothers. Recently, the smoothing properties of SIMPLE-type methods have been studied more carefully for the Stokes problem and a new pressure Schur complement scheme with better smoothing properties has been proposed by Braess and Sarazin [3]. These new type of smoothers will be compared with the Vanka-type smoothers for the parallel solution of the incompressible Navier–Stokes equations.

The current numerical studies are based on

- the non-conforming P_1/P_0-finite element spatial discretization with upwind stabilization,
- the Crank–Nicolson time discretization,
- unstructured grids,
- a MIMD parallel computer.

The paper is organized as follows. In Section 2 the problems and their discretization are given. Section 3 contains details on the coupled multigrid methods that are used in the numerical tests, including remarks on their general behaviour on parallel computers. Both classes of smoothers are described in Section 4. In this section, algorithmic aspects, like the choice of parameters, are also discussed. The numerical comparison of the two types of smoothers is presented in Section 5. The numerical tests include benchmark problems defined in Reference [1], high Reynolds number lid-driven cavity problems, and the flow through a Venturi pipe. The conclusions obtained in these studies are summarized in Section 6.

2. THE PROBLEMS AND THEIR DISCRETIZATION

We consider the steady state incompressible Navier–Stokes equations

$$\begin{aligned} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} &\quad \text{in } \Omega \\ \nabla\cdot\mathbf{u} = 0 &\quad \text{in } \Omega \\ \mathbf{u} = \mathbf{g} &\quad \text{on } \partial\Omega \end{aligned} \tag{1}$$

and the time-dependent incompressible Navier–Stokes equations

$$\begin{aligned}
&\frac{\partial \mathbf{u}}{\partial t} - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p = \mathbf{f} && \text{in } \Omega\times(0, T]\\
&\nabla\cdot\mathbf{u} = 0 && \text{in } \Omega\times(0, T]\\
&\mathbf{u} = \mathbf{g} && \text{on } \partial\Omega\times(0, T]\\
&\mathbf{u} = \mathbf{u}_0 && \text{in } \Omega \text{ for } t = 0
\end{aligned} \tag{2}$$

In Equations (1) and (2), Ω denotes a bounded domain in $\mathbb{R}^2$ with boundary $\partial\Omega$, $\mathbf{u}$ is the velocity, p is the pressure, ν is the kinematic viscosity of the fluid, $\mathbf{u}_0$ is an initial velocity, $\mathbf{g}$ is a Dirichlet boundary condition satisfying the compatibility condition

$$\int_{\partial\Omega} \mathbf{g}\cdot\mathbf{n}_{\partial\Omega}\,\mathrm{d}\gamma = 0$$

T is a fixed end of the time interval, and $\mathbf{f}$ represents exterior forces.

In the stationary case, the non-linear equation (1) is linearized by a fixed point iteration. In each non-linear iteration step, a so-called Oseen equation

$$\begin{aligned}
&-\nu\Delta\mathbf{u}^{n+1} + (\mathbf{u}^{n}\cdot\nabla)\mathbf{u}^{n+1} + \nabla p^{n+1} = \mathbf{f} && \text{in } \Omega\\
&\nabla\cdot\mathbf{u}^{n+1} = 0 && \text{in } \Omega\\
&\mathbf{u}^{n+1} = \mathbf{g} && \text{on } \partial\Omega
\end{aligned} \tag{3}$$

with the given iterate $(\mathbf{u}^n, p^n)$ having to be solved. Problem (3) is discretized by the non-conforming P_1/P_0-finite element discretization from Crouzeix and Raviart [9]. Let τ_h be an admissible triangulation of Ω into triangles. If necessary, the boundary $\partial\Omega$ is approximated by a polygon with the vertices of the polygon on $\partial\Omega$. Then, the discrete velocity is computed as an element of

$$V_h := \left\{\begin{matrix}\text{space of vector-valued piecewise linear functions that are}\\ \text{continous at the midpoints of edges of the triangulation}\end{matrix}\right\}$$

whereas the discrete pressure belongs to the space

$$Q_h := \{\text{space of piecewise constant functions with zero mean value on } \Omega\}$$

This pair of non-conforming finite element spaces guarantees the inf–sup stability condition uniformly with respect to the mesh size and the shape regularity constant of the mesh. Besides this favourable analytical property, there are also advantages from the point of view of the implementation on a parallel computer. The degrees of freedom of the velocity are connected with the midpoints of the edges of the triangles (see Figure 1). Thus, those degrees of freedom that belong to an interface between two sub-domains, each of which stored on a separate processor, have to be stored on two processors only. That is why only one communication is needed to interchange information between the same degrees of freedom on different processors. This is, in general, not true for conforming finite element approximations, where the degrees of freedom are connected with the vertices of triangles. In this case, the degrees of freedom at so-called cross-points can belong to and have to be stored on more than two processors. As a consequence, a complete interchange of information in case of cross-points requires more than one communication. This results in an increase of the communication overhead.

If convection dominates, an additional stabilization becomes necessary. We use a Samarskij upwinding stabilization analysed by Schieweck and Tobiska [10] for the non-conforming P_1/P_0-finite element discretization of the steady state Navier–Stokes equations.

In addition, for improving the accuracy of the discrete solution, we apply the so-called pressure separation technique proposed by Schieweck [6] and Dorok [11]. Instead of the original problem (1), we solve

$$\begin{aligned} -\nu\Delta\mathbf{u}+(\mathbf{u}\cdot\nabla)\mathbf{u}+\nabla(p-\tilde{p}) &= \mathbf{f}-\nabla\tilde{p} && \text{in } \Omega \\ \nabla\cdot\mathbf{u} &= 0 && \text{in } \Omega \\ \mathbf{u} &= \mathbf{g} && \text{on } \partial\Omega \end{aligned} \tag{4}$$

with a suitable approximation $\tilde{p}$ of p. The idea behind this technique is to reduce the part of the error in the error estimate for (1) that is caused by the term $ch\,Re\|p\|_1$, where h is the maximal diameter of all triangles of τ_h and Re is the Reynolds number, which is proportional to ν^{-1}. Thus, if $\|p-\tilde{p}\|_1 \ll \|p\|_1$, the negative influence of increasing the Reynolds number on the accuracy becomes less dramatic. There are several suggestions for choosing $\tilde{p}$ [12]. In our numerical studies, we solve Equation (4) iteratively and use a projection of the current non-conforming approximation of p onto a conforming function $\tilde{p}$ to determine the next approximate of p. In a different context, the same projection has been used by Oswald [13].

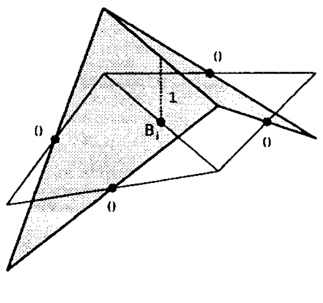

Figure 1. Scalar component of a basis function of V_h.

Now we describe the temporal discretization for discretizing Equation (2). Let t_k and t_{k+1} be two successive discrete times, $\mathbf{u} = \mathbf{u}(t_{k+1})$, $p = p(t_{k+1})$, $\tau = t_{k+1} - t_k$, and $\mathbf{f}_1(t_k) = \nu\Delta\mathbf{u}(t_k) - (\mathbf{u}(t_k)\cdot\nabla)\mathbf{u}(t_k) + \mathbf{f}(t_k)$. We apply a Crank–Nicolson scheme of the form

$$\begin{aligned}
&\frac{\mathbf{u}-\mathbf{u}(t_k)}{\tau} + \frac{1}{2}(-\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u}) + \nabla p = \frac{1}{2}\mathbf{f}(t_{k+1}) + \frac{1}{2}\mathbf{f}_1(t_k) && \text{in } \Omega\\
&\nabla\cdot\mathbf{u} = 0 && \text{in } \Omega\\
&\mathbf{u} = \mathbf{g}(t_k + 1) && \text{on } \partial\Omega
\end{aligned} \tag{5}$$

In this scheme, the velocity and the external forces of the previous time step but not the pressure are used to compute the right-hand side. The term ∇p may be replaced by $(\nabla p + \nabla p(t_k))/2$. Turek [4] stated that both strategies lead to results with the same accuracy. The advantage of Equation (5) consists in not needing to store $p(t_k)$. The linearization and discretization in space is carried out analogously to the steady state problem. In a comparative study [12], the Crank–Nicolson scheme (5) has produced often more accurate results than the BDF (2) scheme.

Thus, the linearization and discretization of the incompressible Navier–Stokes equations lead to large saddle point problems of the following form:

$$\mathscr{A}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \tag{6}$$

3. COUPLED MULTIGRID METHODS

Coupled multigrid methods compute the solution for both types of unknowns (velocity and pressure) of Equation (6) simultaneously. Here, we describe those components of the multigrid methods that are not varied in this paper, in particular the grid transfer operations, the coarsest grid solver, and a step length control. A detailed description of the different types of smoothers is given in Section 4.

3.1. The prolongation and the coarser grid systems

We use L^2-projections for the prolongation. Within the multigrid method, a linear system has to be defined on each level l, $0 \le l \le l_{\max}$. The matrix $\mathscr{A}_l$ is assembled by discretizing the Oseen problem (6) on level l. The convection on level l is given by the restriction of $\mathbf{u}^n$ from the finest level to level l with L^2-projections. The right-hand side of level l is defined by defect restriction, i.e. by testing the defect of level $l+1$ with the prolongation of all basis functions of the finite element space on level l. Note, the defect restriction and the restriction of the finite element function $\mathbf{u}^n$ to level l are different operations.

3.2. The coarsest grid solver

The linear system on the coarsest grid is often too large for an efficient direct solution. Therefore, we use an iterative method, namely the same iterative method used as the smoother on the finer levels.

3.3. The step length control

In addition to standard multigrid methods we apply a step length control after each multigrid cycle. Let $(u,p)^i$ be the current iterate and $(\delta u, \delta p)^{i+1}$ be the correction proposed by the multigrid method. This correction is accepted only as a direction of correction and the next iterate is set to be

$$\begin{pmatrix} u \\ p \end{pmatrix}^{i+1} = \begin{pmatrix} u \\ p \end{pmatrix}^{i} + \kappa \begin{pmatrix} \delta u \\ \delta p \end{pmatrix}^{i+1}$$

The factor $\kappa \in \mathbb{R}$ is chosen such that the Euclidean norm of the residual of the linearized system (6) becomes minimal. This one-dimensional optimization problem has the solution

$$\kappa = \frac{\left(\begin{pmatrix} f \\ g \end{pmatrix} - \mathscr{A} \begin{pmatrix} u \\ p \end{pmatrix}^{i} \right) \cdot \left(\mathscr{A} \begin{pmatrix} \delta u \\ \delta p \end{pmatrix}^{i+1} \right)}{\left(\mathscr{A} \begin{pmatrix} \delta u \\ \delta p \end{pmatrix}^{i+1} \right) \cdot \left(\mathscr{A} \begin{pmatrix} \delta u \\ \delta p \end{pmatrix}^{i+1} \right)}$$

Numerical tests [12] show the important role of the step length control to improve the efficiency of multigrid methods.

3.4. Remarks on the parallel behaviour

Multigrid methods show in many situations a good numerical efficiency. However, their parallel efficiency is not always convincing. The reason for this behaviour can be detected on all grids that are coarser than the finest one. The numerical work (flops) on these grids is, in general, small compared with the amount of communications. Even idleness of processors will occur. This situation is worst on the coarsest grid.

Using the V-cycle, the losses of parallel and total efficiency (computing time) are less dramatic provided that the problems are large (with respect to the number of processors) and the number of levels is moderate (≤ 10), e.g. see McBryan *et al.* [14].

Studies of the V-cycle and deeply refined multigrids were done by Axelsson and Neytcheva [15], who introduced the short multigrid cycle. In the short multigrid cycle, the level of the coarsest grid is defined as a function of the level of the finest grid. The coarsest grid in the short multigrid cycle possesses enough degrees of freedom such that the ratio of numerical work to communications is much better than on the original coarsest grid. In addition, idleness of processors is avoided.

The application of the W-cycle for improving the numerical efficiency and robustness can lead to a considerable loss of parallel and total efficiency on parallel computers. In order to enhance the parallel and total efficiency of the W-cycle, the coarsest grid plays an essential role. Either it can be defined as a function of the finest grid, like in the short multigrid cycle, or the iterations on the coarsest grid should be limited to a small number. In addition, we do not compute the norm of the residual before, during and after the iteration on the coarsest grid since this computation requires a global communication each time. The gain of parallel as well as total efficiency of this approach was demonstrated in Reference [16].

The number of grid levels is moderate in all numerical studies presented in Section 5. Using the V-cycle, we solve the coarsest grid systems iteratively up to a reduction of the Euclidean norm of the residual by the factor 10. In the W-cycle, only ten iteration steps are performed on the coarsest grid without computing the norm of the residual. The F-cycle showed in our tests with both strategies a similar behaviour. We present in this paper the results for the F-cycle, which were obtained with ten iterations on the coarsest grid avoiding the computation of the norm of the residual.

4. THE SMOOTHERS

The main objective of the paper is to compare different iterative schemes as smoothers within coupled multigrid methods. On each level of a coupled multigrid method, a system of form (6) has to be solved approximately. The smoother should damp out the highly oscillating error modes of these systems. In the following, we shall omit the indices for indicating the levels.

4.1. The Vanka-type smoothers

The Vanka-type smoothers, originally proposed by Vanka [2] for finite difference schemes, can be considered as block Gauss–Seidel methods, where a block corresponds to all degrees of freedom that are connected with one element. For the non-conforming P_1/P_0-finite element discretization, these are six velocity degrees of freedom and one pressure degree of freedom (see Figure 2). Thus, a smoothing step with a Vanka-type smoother consists in a loop over all elements, where in each element a 7×7 linear system of equations has to be solved. The

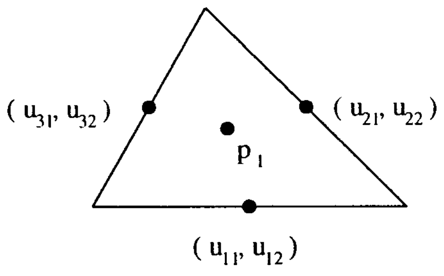

Figure 2. Degrees of freedom in the Vanka-type smoothers.

degrees of freedom are updated in a Gauss–Seidel manner. Since each velocity degree of freedom belongs to two triangles, it is updated twice in one smoothing step.

We denote by $\mathscr{A}_T$ the block of the matrix $\mathscr{A}$ that is connected with the degrees of freedom of the element T, i.e. the intersection of the rows and columns of $\mathscr{A}$ with the global indices of $\{u_{11}, \ldots, u_{32}, p_1\}$,

$$\mathscr{A}_T = \begin{pmatrix} A_T & B_T \\ B_T^T & 0 \end{pmatrix} \in \mathbb{R}^{7\times 7}$$

In addition, we define

$$\mathscr{D}_T = \begin{pmatrix} \operatorname{diag}(A_T) & B_T \\ B_T^T & 0 \end{pmatrix} \in \mathbb{R}^{7\times 7}$$

Similarly, we denote by $(\cdot)_T$ the restriction of a vector on the rows corresponding to the degrees of freedom connected with T.

Diagonal Vanka smoother
The diagonal Vanka smoother updates the velocity and pressure values connected to element T by

$$\begin{pmatrix} u \\ p \end{pmatrix}_T := \begin{pmatrix} u \\ p \end{pmatrix}_T + \mathscr{D}_T^{-1}\left(\begin{pmatrix} f \\ g \end{pmatrix} - \mathscr{A}\begin{pmatrix} u \\ p \end{pmatrix}\right)_T$$

Full Vanka smoother
The full or stabilized Vanka smoother computes new velocity and pressure values in each element by

$$\begin{pmatrix} u \\ p \end{pmatrix}_T := \begin{pmatrix} u \\ p \end{pmatrix}_T + \mathscr{A}_T^{-1}\left(\begin{pmatrix} f \\ g \end{pmatrix} - \mathscr{A}\begin{pmatrix} u \\ p \end{pmatrix}\right)_T$$

This smoother was found to be superior to the diagonal Vanka smoother on anisotropic meshes (Reference [6], Remark 8.1). The full Vanka smoother needs three times more floating point operations than the diagonal Vanka smoother. However, in numerical tests we have observed an increase of the computing time by a factor of about 1.5 only. The full Vanka smoother takes advantages from the current hardware architecture, in particular from very fast floating point operations on data that are stored on the cache memory. The computing times of the diagonal Vanka and the full Vanka smoother are considerably determined by the memory access times for loading subsequently the data of the elements. These memory accesses are time-consuming in comparison with the floating point operations in the cache.

Because the Vanka-type smoothers solve elementwise saddle point problems of form (6), they are also called local smoothers ('local MPSC' in Reference [5]). In Reference [5], the behaviour of Vanka-type smoothers is investigated in a non-parallel context and for the non-conforming rotated Q_1/Q_0-finite element discretization.

In order to avoid communications within a smoothing iteration, we apply the Vanka-type smoothers in parallel on each processor. The values at the interfaces are averaged by one communication step after each smoothing iteration.

Since the Vanka-type smoothers are block Gauss–Seidel methods, the numbering of the degrees of freedom can considerably influence their behaviour. The Vanka-type smoothers are applied by a loop over all elements. We have used lexicographical ordering: an element with barycentre co-ordinates (x_0, y_0) is predecessor of an element with barycentre co-ordinates (x_1, y_1) iff $(x_0 < x_1) \vee ((x_0 = x_1) \wedge (y_0 < y_1))$.

4.2. The Braess–Sarazin-type smoothers

Braess and Sarazin [3] studied pressure Schur complement schemes as smoothers in coupled multigrid methods for the Stokes equations. A well-known pressure Schur complement scheme is the SIMPLE algorithm by Patankar and Spalding [17]. This algorithm applied to solve (6) has the form

SIMPLE algorithm

1. Given p^0, $\alpha > 0$ and C^{-1} which is an approximation to A^{-1}.
2. Iterate $k=0, \ldots, K-1$

$$p^{k+1}=p^k+q^{k+1}=p^k-\alpha(B^TC^{-1}B)^{-1}(B^TA^{-1}(Bp^k-f)+g) \tag{7}$$

3. Compute the new velocity

$$u^K=A^{-1}(f-Bp^K)-(\alpha C)^{-1}Bq^K \tag{8}$$

For given α and C^{-1}, the error in iteration (7) depends only on the current pressure iterate p^k but not on any velocity value. Therefore, this algorithm is called p-dominant in Reference [3]. It is well known that SIMPLE converges if α is chosen small enough. Analytical investigations as well as numerical tests in Reference [3] showed that the smoothing property of the SIMPLE algorithm may be poor and therefore it is in general not suited as a smoother.

In Reference [3], a pressure Schur complement scheme was proposed, which shows better smoothing properties than SIMPLE. Iteration (7) and the velocity update (8) can be written in the form

$$p^{k+1}=p^k-\alpha(B^TC^{-1}B)^{-1}(B^TA^{-1}(Au^k+Bp^k-f)+(g-B^Tu^k))$$

$$u^K=A^{-1}(f-Au^k-Bp^K)-(\alpha C)^{-1}Bq^K+u^k$$

for an arbitrary velocity u^k. Now, the new pressure Schur complement scheme, in the following called the Braess–Sarazin-type smoother, can be derived from the SIMPLE algorithm by replacing A^{-1} with $(\alpha C)^{-1}$. Thus, it has the form

Braess–Sarazin-type smoother

```
1. Given $u^0$, $\alpha > 0$ and $C^{-1}$.
2. Compute the new pressure
```

$$p^{k+1} = p^k + q^{k+1} = p^k - (B^TC^{-1}B)^{-1}(B^TC^{-1}(Au^k + Bp^k - f) + \alpha(g - B^Tu^k))$$
$$= -(B^TC^{-1}B)^{-1}(B^TC^{-1}(Au^k - f) + \alpha(g - B^Tu^k)) \qquad (9)$$

```
3. Compute the new velocity
```

$$u^{k+1} = u^k - (\alpha C)^{-1}((Au^k + Bp^k - f) + Bq^{k+1}) = u^k - (\alpha C)^{-1}(Au^k + Bp^{k+1} - f) \qquad (10)$$

```
4. If stopping criterion is fulfilled then stop, else go to 2.
```

Note, this algorithm can be written in the form

$$\begin{pmatrix} \alpha C & B \\ B^T & 0 \end{pmatrix}\begin{pmatrix} u^{k+1} \\ p^{k+1} \end{pmatrix} = \begin{pmatrix} f - (A - \alpha C)u^k \\ g \end{pmatrix} \qquad (11)$$

It differs in some important properties from SIMPLE. First, the error in the Braess–Sarazin-type smoothers depends only on the velocity u^k but not on the pressure. Therefore, these smoothers are called u-dominant. Second, if Equations (9) and (10) are solved exactly, then u^{k+1} satisfies $B^Tu^{k+1} = g$. Third, the convergence of the coupled multigrid W-cycle with Braess–Sarazin-type smoothers was proven for the Stokes equations, $C = I$, and inf–sup stable conforming finite element methods if α is chosen large enough [3]. This proof can be extended to the non-conforming P_1/P_0-finite element discretization [18].

In contrast to the Vanka-type smoothers, the Braess–Sarazin-type smoothers are non-local, i.e. a linear saddle point problem of form (11) for all degrees of freedom has to be solved. For this reason, they are called global smoothers.

The numerical behaviour of four different SIMPLE-like pressure Schur complement schemes in coupled multigrid methods applied to the stationary Navier–Stokes equations has been studied in Reference [8]. Three of these methods behaved similarly, whereas the fourth was not robust in some test cases. It was already mentioned [8] that SIMPLE-type methods might not be as efficient smoothers as local coupled methods like Vanka-type smoothers. The numerical tests presented in Section 5, in which the efficiency is measured by computing times, will maintain this statement also for Braess–Sarazin-type smoothers. However, compared with the results obtained in Reference [7], the pressure Schur complement schemes behaved much better as smoothers in coupled multigrid methods than as solvers for (6).

The approximation C^{-1} of A^{-1} must be chosen for the Braess–Sarazin-type smoothers. In the case of the Stokes problem, the matrix A is symmetric positive definite and Braess and Sarazin [3] have obtained good results with $C^{-1} = I$ and even better results with the preconditioning matrix for A, which is usually used as a symmetric successive overrelaxation (SSOR) preconditioner [19]. For the steady state Navier–Stokes equations, A is no longer symmetric, therefore we use $C^{-1} = [\mathrm{ILU}_\beta(0)(A)]^{-1}$, i.e. a block ILU-decomposition of A with no fill-in and the fixed damping parameter $\beta = 1$ [20,21]. In this decomposition, a block corresponds to all velocity degrees of freedom that are stored on one processor and within the blocks a lexicographical ordering of the degrees of freedom is used.

For the time-dependent Navier–Stokes equations, discretized with small time steps, the matrix A is close to a diagonal matrix. In this case, we can choose $C^{-1} = (\text{diag}(A))^{-1}$. This choice, together with the use of the non-conforming P_1/P_0-finite element discretization, offers an easy and efficient way to store the Schur complement matrix $B^T C^{-1} B$ explicitly [12]. In this way, the computing times for matrix–vector products can be considerably reduced compared with the successive multiplication with each factor.

An important parameter is the damping factor α. Even for the Stokes equations, Sarazin [19] observed 'small deviations from the optimal value may lead occasionally to very bad rates of convergence'. The dependence of the efficiency of coupled multigrid methods on α for the Navier–Stokes equations is studied numerically in Section 5.

For the choice $C^{-1} = [\text{ILU}_\beta(0)(A)]^{-1}$, the computation of the new pressure (9) requires the solution of the linear system of equations

$$B^T[\text{ILU}_\beta(0)(A)]^{-1} B p^{k+1} = -B^T[\text{ILU}_\beta(0)(A)]^{-1}(Au^k - f) - \alpha(g - B^T u^k) \qquad (12)$$

The system matrix of (12) is known only implicitly. Therefore, iterative schemes that require only matrix–vector products seems to be appropriate solvers of (12). Since A is in general non-symmetric, $\text{ILU}_\beta(0)(A)$ and $B^T[\text{ILU}_\beta(0)(A)]^{-1}B$ will be non-symmetric, too. The numerical costs of a smoothing step depend on the number of iterations for the solution of (12). We want to apply only a few iterations in order to prevent a very large increase of these costs. That is why iterative solvers with unpredictable behaviour at the beginning of the iteration (e.g. CGS) have not been taken into consideration. We employ instead a GMRES from Saad and Schultz [22]. This solver can be applied to non-symmetric systems and the norm of the residual is reduced in each iteration. However, additional memory requirements are connected with GMRES of about the number of pressure degrees of freedom on the finest level times the number of GMRES iterations (the last number is chosen small to avoid a restart).

In the case $C^{-1} = (\text{diag}(A))^{-1}$, the new pressure is computed by solving

$$B^T(\text{diag}(A))^{-1} B p^{k+1} = -B^T(\text{diag}(A))^{-1}(Au^k - f) - \alpha(g - B^T u^k) \qquad (13)$$

where $B^T(\text{diag}(A))^{-1}B$ is stored explicitly. The system matrix in (13) is symmetric and positive definite after fixing the constant of the pressure. We solve (13) with a preconditioned conjugate gradient (PCG) method with diagonal preconditioner. The easy availability of this preconditioner results from the explicit storage of $B^T(\text{diag}(A))^{-1}B$.

5. NUMERICAL STUDIES

All computations were performed on a Parsytec GCPowerPlus (80 MHz, 9.2 MFlops/processor (LINPACK), 35 MB s^{-1} communication, 5 μs message set-up time, 60 μs minimal network latency). This parallel computer has a fast floating point operation speed compared with its communication speed. The implementation of the algorithms was done using parts of the program package *ugp*1.0 by Bastian and co-workers [23]. This program is an early version of the now available program *UG*3 [24]. In particular, data structures, load balancing routines and the parallel environment of *ugp*1.0 were used. However, the data structures had to be extended in order to handle multi-dimensional data and non-conforming finite elements.

Next we explain the abbreviations in the tables given below:

diagVan	diagonal Vanka smoother
fullVan	full Vanka smoother
BS_ilu	Braess–Sarazin-type smoother with $C^{-1} = [\mathrm{ILU}_1(0)(A)]^{-1}$ and GMRES as solver for (12)
BS_diag	Braess–Sarazin-type smoother with $C^{-1} = (\mathrm{diag}(A))^{-1}$ and PCG as solver for (13)

On each level, which is not the coarsest one, pre- and postsmoothing iterations are applied with the same number n_{sm} of smoothing steps. This number was chosen equal for all levels. In the stationary test problems, we used as initial guess for the fixed point iteration on level l the interpolation of the solution on level $l-1$. In the time-dependent case, a good initial guess for the fixed point iteration at time step t_{k+1} is easily available by using the solution of the previous time step t_k. The linear saddle point problem (6) was solved in each step of the fixed point iteration up to a reduction of the Euclidean norm of the residual by the factor 10.

The different smoothers are compared with respect to the total computing time. Although the computing time depends on the hardware, the programming language, the data structures, the compiler, etc., it is the most important measure in applications. Since all algorithms are implemented in the same code, the above mentioned aspects should have approximately the same influence on all methods. Thus, we think that a comparison by computing times gives fair evidence about the capabilities of each method.

5.1. Example 1. A steady state benchmark problem

Numerical studies were performed on a benchmark problem for the steady state Navier–Stokes equations defined in the DFG priority research program ‘Flow simulation with high-performance computers’ [1]. The problem describes the flow in a channel around a cylinder, which is slightly closer to the lower than to the upper wall (see Figure 3).

The solution of (1) has to be computed with $\nu = 10^{-3}$, $\mathbf{f} = 0$, a parabolic inflow and outflow

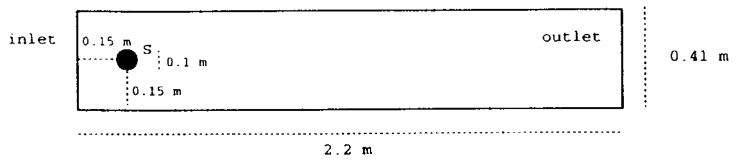

Figure 3. Domain of the DFG-benchmark problems.

profile with maximal value 0.3, and no-slip conditions on the other boundaries. The Reynolds number of this flow is $Re = 20$.

The coarsest grid (level 0) shown in Figure 4 was used. We present results for refinement level 4 (296832 velocity degrees of freedom, 99328 pressure degrees of freedom). The stopping criterion for the fixed point iteration was an Euclidean norm of the residual lower than 10^{-10}. The parallel computations were carried out on eight processors.

The results of the computational studies are presented in Tables I and II. The Vanka-type smoothers proved to be clearly more efficient in these studies than the Braess–Sarazin-type smoother.

Employing the Vanka-type smoothers, the best choices were two to four smoothing steps for all types of multigrid cycles (see Table I). Within this range, the full Vanka V-cycle was the best solver. For a small number of smoothing steps, the full Vanka smoother was in general somewhat superior to the diagonal Vanka smoother. The W(1, 1)-cycle showed bad behaviour. It converged very slowly for the full Vanka smoother and did not converge for the diagonal Vanka smoother. The behaviour of the diagonal Vanka F(1, 1)-cycle was also unsatisfactory.

Table II presents the computing times and the communication overheads for the Braess–Sarazin-type smoother with different combinations of the number of smoothing steps and the number of GMRES iterations for solving (12). The damping factor was set to be $\alpha = 1.0$. The

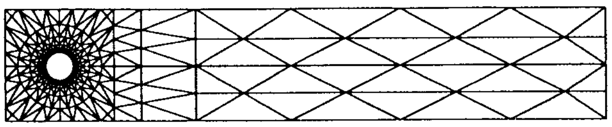

Figure 4. Coarsest grid for the steady state benchmark problem.

Table I. Steady state benchmark problem, Vanka-type smoothers.

Cycle	Smoother	n_{sm}	1	2	3	4	6	8	10
V	diagVan	Comp. time	1103	1056	910	928	1025	1144	1205
		Comm. in %	4.6	4.6	3.5	3.4	3.1	2.9	2.8
	fullVan	Comp. time	815	715	741	844	1248	1217	1283
		Comm. in %	3.1	2.6	2.5	2.4	2.3	1.8	1.6
F	diagVan	Comp. time	2065	764	803	916	962	1131	1319
		Comm. in %	3.7	3.0	2.7	2.6	2.4	2.4	2.3
	fullVan	Comp. time	883	753	873	1157	1073	1108	1404
		Comm. in %	2.8	2.2	2.0	1.9	1.8	1.7	1.7
W	diagVan	Comp. time	div.	793	841	969	975	1151	1345
		Comm. in %	—	4.5	4.0	3.7	3.4	3.3	3.2
	fullVan	Comp. time	4087	762	754	908	1093	1168	1286
		Comm. in %	4.5	3.4	2.7	2.5	2.3	2.5	2.4

Table II. Steady state benchmark problem, Braess–Sarazin-type smoother, $C^{-1} = [\mathrm{ILU}_1(0)(A)]^{-1}$.

Cycle	n_{sm}	GMRES iter.	3	5	7	10	12	15	20
V	1	Comp. time	4271	3144	2512	1997	1995	1977	2478
		Comm. in %	8.2	9.3	11.6	11.6	12.2	12.8	13.4
	2	Comp. time	3066	2384	1987	1806	1957	2185	2803
		Comm. in %	8.9	9.3	9.4	9.4	11.3	11.8	12.3
	3	Comp. time	2964	2243	1900	1900	2199	2645	3426
		Comm. in %	7.9	8.4	8.9	8.5	9.2	9.4	10.3
F	1	Comp. time	10720	11090	7583	8065	8921	12140	38490
		Comm. in %	10.2	12.6	14.6	17.6	19.5	22.0	25.6
	2	Comp. time	3458	2725	2259	2061	2243	2532	3649
		Comm. in %	7.1	8.6	10.1	12.0	13.2	15.1	17.9
	3	Comp. time	3347	2608	2149	2430	2836	3474	4449
		Comm. in %	6.4	7.7	9.0	10.8	11.7	13.5	17.0
W	1	Comp. time	13460	15180	10320	9790	10590	22710	div.
		Comm. in %	15.6	19.0	22.0	25.2	27.6	30.9	—
	2	Comp. time	3813	2983	2461	2330	2556	2963	4019
		Comm. in %	11.8	14.2	16.1	19.1	20.9	24.0	28.1
	3	Comp. time	3669	2919	2414	2755	3247	4008	5155
		Comm. in %	11.5	13.6	15.4	17.8	19.6	21.9	25.2

V-cycle was the most efficient choice for this smoother. In our tests, the best results were obtained with the V(2, 2)-cycle and ten GMRES iterations. However, the fastest computing time with the Vanka-type smoothers is about 2.5 times better. For all types of cycles we observe that the optimal number of GMRES iterations decreases if the number of smoothing steps increases. The F(1, 1) and W(1, 1)-cycle had problems to converge also with the Braess–Sarazin-type smoother.

For the V(2, 2)-cycle with ten GMRES iterations, we studied also the dependence of the computing time on the damping factor α (see Table III). The minimal computing time was obtained with $\alpha = 1.25$, but the computing times varied only slightly for $\alpha \in [0.75, 1.5]$. In this range, the Braess–Sarazin-type smoother showed only a weak dependence on α. If α was chosen too small, e.g. $\alpha = 0.5$, the multigrid method did not converge. In References [3,18] it was proven that the coupled multigrid method with the Braess–Sarazin-type smoothers applied to the Stokes equations converges if α is large enough. The same behaviour can be seen now numerically for the Navier–Stokes equations.

The lower efficiency of the Braess–Sarazin-type smoother is not caused by its smoothing property since the number of multigrid cycles to solve the linear systems was in general not greater than for the Vanka-type smoothers. The main reason for the larger computing times are the much higher computational costs to perform a smoothing step.

It was observed in Reference [7] that pressure Schur complement schemes for the solution of Navier–Stokes equations need considerable more communications on parallel computers than coupled multigrid methods with Vanka-type smoothers. Comparing the communication overhead of the coupled multigrid methods with the different smoothers, a similar observation

Table III. Steady state benchmark problem, Braess–Sarazin-type smoother, $C^{-1} = [\mathrm{ILU}_1(0)(A)]^{-1}$, V(2, 2)-cycle, dependence on α.

α	0.5	0.75	0.8	1.0	1.25	1.5	2.0	3.0	4.0
	div.	1856	1859	1806	1757	1978	2500	3519	5001

can be made. While the Vanka-type smoothers had a communication overhead of 1.6–4.6 per cent of the computing time, the Braess–Sarazin-type smoother needed 6.4–30.9 per cent. The latter may be improved somewhat, for instance, by using an iterative method on the coarsest grid that needs less communications than the Braess–Sarazin-type smoother.

5.2. Example 2. Lid-driven cavity flow

We study the lid-driven cavity problem for different Reynolds numbers $Re = \nu^{-1}$. This problem is defined in the unit square. The right-hand side $\mathbf{f}$ is set to be zero. No-slip boundary conditions are imposed for $x = 0$, $x = 1$, and $y = 0$. The velocity at the upper boundary is given by $\mathbf{g} = (1, 0)$.

Figure 5 presents the initial triangulation of the domain (level 0). The studies were performed on level 7 (392192 velocity degrees of freedom, 131072 pressure degrees of freedom) and the fixed point iteration was stopped for an Euclidean norm of the residual lower than 10^{-10}. The computations were carried out on eight processors.

The results of the numerical studies are presented in Table IV. In these studies, the Braess–Sarazin-type smoother was applied with $\alpha = 1$ and ten GMRES iterations. As in the previous example, the Vanka-type smoothers were clearly superior to the Braess–Sarazin-type smoother. The fastest solution times were obtained with the diagonal Vanka smoother applied in the F- and W-cycles. The more expensive full Vanka smoother did in general not improve the behaviour of the solver. The V-cycle turned out to be inefficient. The most important reason of the high computing times of the Braess–Sarazin-type smoother within the W-cycle is again the enormous parallel overhead.

5.3. Example 3. A time-dependent benchmark problem

Within the DFG priority research program 'Flow simulation with high-performance computers' [1], time-dependent benchmark problems describing flows in the channel depicted in Figure 3 were defined. We consider a Kármán vortex street given by $\nu = 10^{-3}$, $\mathbf{f} = 0$, a steady state parabolic inflow and outflow profile with maximal value 1.5, and no-slip conditions on the other boundaries. The length of a period is denoted by T and it was found numerically to be $T \approx 0.335$ s. The Reynolds number of this flow is $Re = 100$.

The coarsest grid shown in Figure 6 was used for the computations. We present results on level 5 (588416 velocity degrees of freedom, 196608 pressure degrees of freedom) (Tables V, VI and VII). The Crank–Nicolson scheme (5) was used with equidistant time steps τ. The stopping criterion for the fixed point iteration in each time step was an Euclidean norm of the

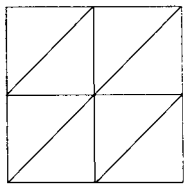

Figure 5. Initial triangulation for the lid-driven cavity problem.

residual less than 10^{-6}. Two pre- and two postsmoothing steps were applied on each level except on the coarsest one. The Braess–Sarazin-type smoothers were applied with ten GMRES or PCG iterations for solving (12) or (13) respectively. The pressure separation was not used in the spatial discretization of this problem because it results in an increase of the computing times by a factor of 2–4. The computations were performed on 16 processors.

Table IV. Lid-driven cavity problem for different values of ν.

Cycle	Smoother	$n_{sm}\backslash\nu^{-1}$	1000	2000	4000	6000
V	diagVan	2	3888	8280	9709	10 860
		3	3485	6422	9578	12 040
		4	3888	8216	9730	12 990
	fullVan	2	3258	6824	11 030	13 800
		3	4067	6570	9905	15 760
		4	3471	7822	13 260	19 120
	BS_ilu	2	5717	10 410	14 710	17 420
		3	4628	9857	16 330	22 220
F	diagVan	2	907	1558	3162	6578
		3	1184	1915	3182	11 010
		4	1344	2227	3904	8231
	fullVan	2	1259	2180	3834	7244
		3	1514	2691	4485	10 310
		4	1663	2955	5405	10 190
	BS_ilu	2	3030	6086	12 760	16 100
		3	3777	6713	16 430	21 670
W	diagVan	2	991	1518	3995	6234
		3	1092	1890	3877	9809
		4	1144	2291	4569	8128
	fullVan	2	1266	2008	4176	8206
		3	1349	2474	4899	11 000
		4	1370	3059	5917	10 640
	BS_ilu	2	4214	10 310	21 370	26 800
		3	5748	8882	26 480	35 490

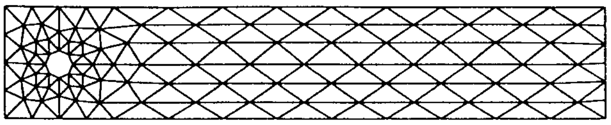

Figure 6. Coarsest grid for the time-dependent benchmark problem.

Table V. Time-dependent benchmark problem, time step $\tau = 0.01$.

Cycle	Smoother	α	Time/period	Time/step	Comm. in %
V	diagVan		3664	102	3.5
	fullVan		3991	111	2.9
	BS_ilu	1.0	9270	258	10.0
	BS_diag	1.5	8033	223	7.4
F	diagVan		2996	83	4.5
	fullVan		3889	108	3.4
	BS_ilu	1.0	12 304	342	22.8
	BS_diag	1.5	10 240	284	16.3
W	diagVan		3669	102	7.0
	fullVan		4409	122	6.1
	BS_ilu	1.0	16 898	469	39.3
	BS_diag	1.5	13 684	380	30.3

The most efficient solver for all time steps was the F-cycle with the diagonal Vanka smoother. The diagonal Vanka smoother outperformed the full Vanka smoother in this example. The V- and F-cycles were superior to the W-cycle for large time steps and for both types of Vanka smoothers. Similar to the stationary benchmark problem, the best computing time with the Vanka-type smoothers is about 2.5 times better than with the Braess–Sarazin-type smoothers.

Using the Braess–Sarazin-type smoothers, the V-cycle was clearly the most efficient cycle. Analogously to the stationary problem, the W-cycle showed an enormous parallel overhead. The choice of $C = \mathrm{diag}(A)$ combined with the explicit storage of $B^T(\mathrm{diag}(A))^{-1}B$ and the solution of (13) with PCG ('BS_diag') resulted in faster solutions of the instationary Navier–Stokes equations with less parallel overhead than choosing $C = [\mathrm{ILU}_\beta(0)(A)]$ and solving (12) with GMRES ('BS_ilu').

The dependence on α of the coupled multigrid method V(2, 2)-cycle with the Braess–Sarazin-type smoother and $C^{-1} = (\mathrm{diag}(A))^{-1}$ is demonstrated in Table VIII. Similar to the steady state benchmark problem (Table III) there was a range on which the method depends relatively weakly on α (here $\alpha \in [1.25, 1.75]$) and the method did not converge in the case of choosing α too small.

Table VI. Time-dependent benchmark problem, time step $\tau = 0.005$.

Cycle	Smoother	α	Time/period	Time/step	Comm. in %
V	diagVan		5759	82	3.6
	fullVan		6424	92	2.6
	B_Silu	1.0	14 500	207	10.2
	BS_diag	1.5	11 497	164	7.4
F	diagVan		4808	69	4.5
	fullVan		6481	93	3.3
	BS_ilu	1.0	16 213	232	22.6
	BS_diag	1.5	15 511	222	15.5
W	diagVan		5749	82	7.4
	fullVan		7622	109	5.5
	BS_ilu	1.0	22 214	317	39.1
	BS_diag	1.5	19 975	285	28.6

Table VII. Time-dependent benchmark problem, time step $\tau = 0.0025$.

Cycle	Smoother	α	Time/period	Time/step	Comm. in %
V	diagVan		9639	70	3.6
	fullVan		10 646	77	2.5
	BS_ilu	1.0	21 949	159	10.3
	BS_diag	1.5	19 306	140	7.0
F	diagVan		7966	58	4.4
	fullVan		10 674	77	3.2
	BS_ilu	1.0	31 965	232	22.6
	BS_diag	1.5	25 699	186	15.3
W	diagVan		9552	69	7.1
	fullVan		10 470	76	5.2
	BS_ilu	1.0	43 783	317	39.0
	BS_diag	1.5	33 731	244	28.7

5.4. Example 4. Time-dependent flow through a Venturi pipe

Flows through a Venturi pipe are studied in References [4,5]. We consider the Venturi pipe shown in Figure 7 with the coarsest triangulation depicted in Figure 8. A parabolic inflow with maximum 1 is prescribed at the inlet. Do-nothing boundary conditions are applied at the right end of the pipe and the end of the small upper channel. All other boundaries possess no-slip conditions. The viscosity was set to be $\nu = 10^{-2}$, which results in a Reynolds number of $Re \approx 500$ [5]. The solution of the Stokes problem was used as initial condition.

Table VIII. Time-dependent benchmark problem, Braess–Sarazin-type smoother with $C^{-1} = (\text{diag}(A))^{-1}$, V(2, 2)-cycle, $\tau = 0.005$, dependence on α.

Damping factor α	1.0	1.25	1.5	1.75	2.0	3.0
Time/period	div.	10 634	11 497	13 325	14 961	21 418
Time/step		152	164	190	214	306

We studied the Venturi pipe flow on refinement level 5 (183296 velocity degrees of freedom, 61440 pressure degrees of freedom) in the time interval $t \in [0, 20]$ and for equidistant time steps $\tau = 0.02$. Thus, the solution at 1000 time steps had to be computed. The fixed point iteration at each discrete time was stopped if the Euclidean norm of the residual was lower than 10^{-6}. All smoothers were applied with two pre- and two postsmoothing steps. We used eight processors of the parallel computer.

The computing times and the communication overheads found in the numerical studies are presented in Table IX. The Braess–Sarazin-type smoothers were applied with $\alpha = 1$ and ten iterations for the solution of (12) or (13) respectively. The diagonal Vanka smoother showed again the best efficiency. The computing time of 'BS_diag' applied in the V-cycle is, in contrast to the previous examples, not much worse than for the Vanka-type smoothers. This type of Braess–Sarazin smoother shows here an acceptable efficiency for the solution of the time dependent problem.

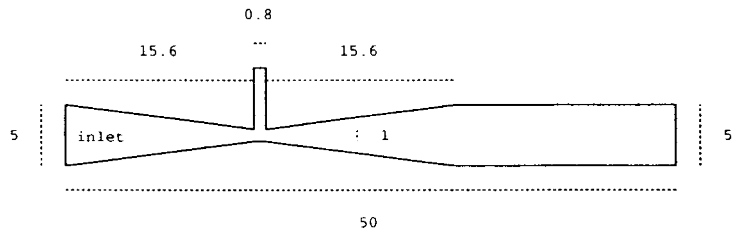

Figure 7. The Venturi pipe.

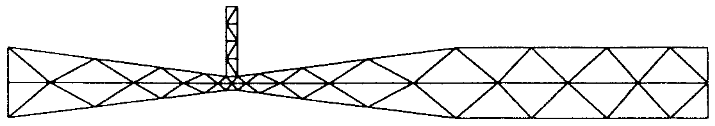

Figure 8. Initial triangulation for the Venturi pipe problem.

Table IX. Flow through the Venturi pipe, $T = 20$, time step $\tau = 0.02$.

Cycle	Smoother	Computing time	Comm. in %
V	diagVan	65 338	6.9
	fullVan	74 671	5.1
	BS_ilu	143 734	19.0
	BS_diag	85 175	9.5
F	diagVan	64 567	6.6
	fullVan	77 600	5.3
	BS_ilu	205 259	27.7
	BS_diag	112 396	15.4
W	diagVan	68 899	8.2
	fullVan	81 138	6.8
	BS_ilu	288 072	38.0
	BS_diag	142 687	24.1

6. SUMMARY

In this paper, two classes of smoothers within coupled multigrid methods for solving two-dimensional incompressible Navier–Stokes equations have been studied. The most important conclusions of the numerical tests are the following:

- The Vanka-type smoothers have clearly been more efficient than the Braess–Sarazin-type smoothers in the steady state problems.
- The Vanka-type smoothers have been superior to the Braess–Sarazin-type smoothers also in the time dependent tests. However, this superiority is not so dramatic as for the steady state problems.
- The F-cycle has always been a good and often even the best choice for the Vanka-type smoothers.
- The inferiority of the Braess–Sarazin-type smoothers is mainly caused by the high computational costs for performing one smoothing step. Moreover, their parallel overhead is very large, especially within the W-cycle.
- The possibility of storing $B^T(\text{diag}(A))^{-1}B$ explicitly in the non-conforming P_1/P_0-finite element discretization has reduced the computing times in the instationary problem as well as the parallel overhead of the Braess–Sarazin-type smoothers considerably.
- There is a sufficiently large interval where the computing times depend only weakly on the damping factor α in the Braess–Sarazin-type smoothers. The coupled multigrid method fails to converge if α is chosen too small.

REFERENCES

1. Schäfer M, Turek S. The benchmark problem 'flow around a cylinder'. In *Flow Simulation with High-Performance Computers II*, vol. 52 of Notes on Numerical Fluid Mechanics, Hirschel EH (ed.). Vieweg: Weisbaden, 1996; 547–566.
2. Vanka S. Block-implicit multigrid calculation of two-dimensional recirculating flows. *Computer Methods in Applied Mechanics and Engineering* 1986; **59**(1): 29–48.
3. Braess D, Sarazin R. An efficient smoother for the Stokes problem. *Applied Numerical Mathematics* 1997; **23**(1): 3–19.
4. Turek S. A comparative study of time-stepping techniques for the incompressible Navier–Stokes equations: from fully implicit non-linear schemes to semi-implicit projection methods. *International Journal for Numerical Methods in Fluids* 1996; **22**: 987–1011.
5. Turek S. *Efficient Solvers for Incompressible Flow Problems: An Algorithmic and Computational Approach*, vol. 6 of Lecture Notes in Computational Science and Engineering. Springer: Berlin, 1999.
6. Schieweck F. Parallele Lösung der stationären inkompressiblen Navier–Stokes Gleichungen. Otto-von-Guericke Universität Magdeburg, Fakultät fur Mathematik, 1997.
7. John V. A comparison of parallel solvers for the incompressible Navier–Stokes equations. *Computing and Visualisation in Science* 2000; to appear.
8. Gjesdal T, Lossius MEH. Comparison of pressure correction smoothers for multigrid solutions of incompressible flow. *International Journal for Numerical Methods in Fluids* 1997; **25**: 393–405.
9. Crouzeix M, Raviart P-A. Conforming and non-conforming finite element methods for solving the stationary Stokes equations I. *RAIRO Analyse Numérique* 1973; **7**: 33–76.
10. Schieweck F, Tobiska L. An optimal order error estimate for an upwind discretization of the Navier–Stokes equations. *Numerical Methods and Partial Differential Equations* 1996; **12**: 407–491.
11. Dorok O. Improved accuracy of a finite element discretization for solving the Boussinesq approximation of the Navier–Stokes equations. In *Proceedings of the Second Summer Conference: Numerical Modelling in Continuum Mechanics (Theory, Algorithms, Applications)*. Prague, 22–25 August, 1994.
12. John V. Parallele Lösung der inkompressiblen Navier–Stokes Gleichungen auf adaptiv verfeinerten Gittern. PhD thesis, Otto-von-Guericke-Universität Magdeburg, Fakultät für Mathematik, 1997.
13. Oswald P. On a hierarchical basis multilevel method with nonconforming P1 elements. *Numerics and Mathematics* 1992; **62**: 189–212.
14. McBryan OA, Frederickson PO, Linden J, Schüler A, Solchenbach K, Stüben K, Thole C-A, Trottenberg U. Multigrid methods on parallel computers-a survey of recent developments. *Impact of Computers in Science and Engineering* 1991; **3**: 1–75.
15. Axelsson O, Neytcheva M. Scalable algorithms for the solution of Navier's equations of elasticity. *Journal of Computer Applications in Mathematics* 1995; **63**: 149–178.
16. John V. On the parallel performance of coupled multigrid methods for the solution of incompressible Navier–Stokes equations. In *Large-Scale Scientific Computations of Engineering and Environmental Problems*, vol. 62 of Notes on Numerical Fluid Mechanics, Griebel M, Iliev OP, Margenov SD, Vassilevski PS (eds). Vieweg: Weisbaden, 1998; 269–280.
17. Patankar SV, Spalding DB. A calculation procedure for heat and mass transfer in three-dimensional parabolic flows. *International Journal for Heat Mass Transfers* 1972; **15**: 1787–1806.
18. John V, Tobiska L. A coupled multigrid method for nonconforming finite element discretizations of the Stokes equation. Preprint 3/99, Fakultät für Mathematik, Otto-von-Guericke-Universität Magdeburg, 1999.
19. Sarazin R. Eine Klasse von effizienten Glättern vom Jacobi-Typ für das Stokes-Problem. PhD thesis, Ruhr-Universität Bochum, 1996.
20. Wittum G. Multi grid methods for Stokes and Navier–Stokes equations. Transforming smoothers: algorithms and numerical results. *Numerics and Mathematics* 1989; **54**: 543–563.
21. Saad Y. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company: Boston, MA, 1996.
22. Saad Y, Schultz MH. GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal of Scientific and Statistical Computing* 1986; **7**(3): 856–869.
23. Bastian P. *Parallele adaptive Mehrgitterverfahren*. Teubner Skripten zur Numerik. Teubner: Leipzig, 1996.
24. Bastian P, Birken K, Johannsen K, Lang S, Neuß N, Rentz-Reichert H, Wieners C. UG—a flexible software toolbox for solving partial differential equations. *Computing and Visualisation in Science* 1997; **1**: 27–40.