On the Parallel Performance of Coupled Multigrid Methods for the Solution of the Incompressible Navier–Stokes Equations

Volker John *

Abstract

This paper presents a detailed numerical study of the parallel performance of coupled multigrid V–, F–, and W–cycle for the solution of the incompressible Navier–Stokes equations. The results show the considerable influence of the number of processors on a good choice of components and parameters of the algorithms. Some general suggestions concerning this choice can be derived from the tests.

Keywords. Incompressible Navier–Stokes equations, coupled multigrid methods, parallel performance, nonconforming finite element methods.

AMS(MOS) subject classifications. 65N30, 65N55, 65Y05

1 Introduction

The fast numerical solution of incompressible Navier–Stokes equations is of importance not only as an autonomous problem but also as a subproblem of more complex problems, e.g. the simulation of crystal growth. Coupled multigrid methods have been proven in a recent study as fast solvers for the discrete saddle point problems coming from the linearization of these equations, see [ST1].

We consider the steady state incompressible Navier–Stokes equations

$$\begin{array}{rcll} -\nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p & = & \mathbf{f} & \text{in } \Omega \\ \nabla\cdot\mathbf{u} & = & 0 & \text{in } \Omega \\ \mathbf{u} & = & \mathbf{g} & \text{on } \partial\Omega \end{array} \qquad (1.1)$$

and the time dependent incompressible Navier–Stokes equations

$$\begin{array}{rcll} \mathbf{u}_t - \nu\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p & = & \mathbf{f} & \text{in } \Omega\times(0,T] \\ \nabla\cdot\mathbf{u} & = & 0 & \text{in } \Omega\times(0,T] \\ \mathbf{u} & = & \mathbf{g} & \text{on } \partial\Omega\times(0,T] \\ \mathbf{u} & = & \mathbf{u}_0 & \text{for } t=0. \end{array} \qquad (1.2)$$

*Otto–von–Guericke–Universität Magdeburg, Institut für Analysis und Numerik, Postfach 4120, D-39016 Magdeburg, Germany, email: volker.john@mathematik.uni-magdeburg.de

In (1.1) and (1.2), Ω is a domain in $\mathbb{R}^2$, $\mathbf{u}$ is the velocity, p is the pressure, ν is the kinematic viscosity of the fluid, $\mathbf{u}_0$ is an initial velocity, and $\mathbf{f}$ represents exterior forces.

Equation (1.1) is linearized using a fix point iteration. In each nonlinear iteration step an Oseen problem is solved which is discretized using the nonconforming P_1/P_0–finite element from Crouzeix and Raviart [CR]. This discretization needs a stabilization in the case of dominant convection. We use Samarskij upwinding which has been analyzed by Schieweck and Tobiska [ST2]. There are two reasons for using the nonconforming P_1/P_0–finite element discretization. First, this pair of discrete spaces guarantees the inf–sup stability condition uniformly with respect to the mesh size and the shape regularity constant of the mesh. Second, the unknowns of the velocity on the interfaces (boundary of a subdomain of Ω which is stored on a processor) have to be stored on two processors only. The reason is that the unknowns of the velocity can be connected to the edges of the triangles. That's why only one communication is sufficient to interchange information between the same unknowns on different processors. This is not true for unknowns which are connected to corners of triangles. In this case there exist some unknowns which are stored on more than two processors, so–called crosspoints.

The time derivative in (1.2) is discretized using a modified Crank–Nicolson scheme which has been proposed by Turek [Tur]. The velocity and the external forces of the previous time step but not the pressure are used to compute the right hand side in the modified Crank–Nicolson scheme. The linearization and discretization in space is carried out analogously to the steady state problem. The modified Crank–Nicolson scheme has produced more accurate results than the BDF(2) scheme in a comparative study, see [Joh].

2 Coupled Multigrid Methods on Parallel Computers

The linearization and discretization of the incompressible Navier–Stokes equations leads to large saddle point problems of the form

$$\mathcal{A}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} A & B \\ B^T & 0 \end{pmatrix}\begin{pmatrix} u \\ p \end{pmatrix} = \begin{pmatrix} r \\ s \end{pmatrix}. \tag{2.1}$$

This system is solved with a coupled multigrid method, i.e. with a multigrid method which computes the results for both types of unknowns (velocity and pressure) simultaneously. An alternative approach are pressure correction schemes where the results for both types of unknowns are computed successively. In a detailed study [Joh] we found coupled multigrid methods much superior to some pressure correction schemes in the steady state case. Even for the time dependent equations the coupled multigrid methods have been faster. This emphasizes the attractivity of these methods.

The efficiency of standard multigrid methods implemented on parallel computers is badly by reason of the grids which are coarser than the finest grid. These grids may be distributed well on all processors. In this case a lot of communication is necessary compared to the relatively small numerical work. The alternative is to concentrate the coarser grids on few processors only. But while these processors are working the others are idle. This problem is most dramatically on the coarsest grid. For that reason the greatest loss on parallel efficiency will occur in multigrid W–cycles which spend a lot of work on coarser grids, especially on the coarsest grid. Thus, to obtain multigrid methods with good parallel efficiency the work spend on the coarsest grid should be small (see below, the alternative approach of the coarsest grid iteration) or even the coarsest grid should be chosen as a function of the finest grid, see Axelsson and Neytcheva [AN].

The grid transfer operations. There are no canonical grid transfer operators since the discrete spaces belonging to the different levels of the multigrid are not nested in the nonconforming finite element discretization. We use L^2–projections for the restriction and prolongation.

The smoothers. The smoother influences the efficiency of multigrid methods considerably. Because of the zero block of $\mathcal{A}$ standard smoothers, like pointwise Gauss–Seidel, cannot be applied to (2.1) without modification. We use the *Vanka smoother*, see e.g. Schieweck [Sch1], which is a blockwise Gauss–Seidel method. A block consists of the unknowns which are connected to a triangle. That are three two–dimensional vectors of the discrete velocity and one unknown of the discrete pressure. Using the Vanka–smoother a (7×7)–system with a matrix of the same structure as $\mathcal{A}$ and a diagonal upper left block must be solved on each element. Schieweck [Sch2] pointed out that for the reason of stability it is sometimes better to use a full upper left block in the (7×7)–systems. We call this smoother *stabilized Vanka smoother*. The solution of its local (7×7)–system needs about three times the numerical work (floating point operations) compared to the Vanka smoother.

On a parallel computer communications within the smoothing process should be avoided. That's why we use a block Jacobi smoother where a block consists of all unknowns which are stored on one processor. In the blocks the Vanka smoother or stabilized Vanka smoother is used. After each smoothing iteration a communication is necessary in order to obtain a consistently stored result via averaging the values.

The coarsest grid solvers. The linear system on the coarsest grid often is too large to solve it efficiently with a direct method. That's why we use an iterative method, namely the same iterative method which is used as smoother on the finer levels. A *standard approach* is to reduce the Euclidean norm of the residual by a prescribe factor. However, this approach has some disadvantages on parallel computers. The numerical work on the coarsest grid is an almost negligible part of the numerical work of the multigrid method. But the parallel work (communications) is not! A communication is necessary after each iteration step, whose number is unknown a priori. A second communication is needed for the computation of the norm of the residual. Thus, in the standard approach the parallel work on the coarsest grid is extremely large compared to the numerical work. An *alternative approach* is to prescribe a small number of iteration steps and break the iteration after these steps without computing the norm of the residual.

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

$$(u^T, p^T)^{i+1} = (u^T, p^T)^i + \kappa(\delta u^T, \delta p^T)^{i+1}.$$

The factor $\kappa \in \mathbb{R}$ is chosen such that the Euclidean norm of the residual is minimal. This one–dimensional optimization problem has the solution

$$\kappa = \frac{((r^T, s^T) - \mathcal{A}(u^T, p^T)^i)^T(\mathcal{A}(\delta u^T, \delta p^T)^{i+1})}{(\mathcal{A}(\delta u^T, \delta p^T)^{i+1})^T(\mathcal{A}(\delta u^T, \delta p^T)^{i+1})}. \tag{2.2}$$

3 The Test Problems

The numerical tests have been performed on benchmark problems of the DFG priority research program "Flow simulation with high–performance computers", compare Schäfer and Turek [ST1]. These problems describe flows in a channel around a cylinder, see Figure 1. The steady state problem is

$$\begin{array}{rcll}
-10^{-3}\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p & = & \mathbf{0} & \text{in } \Omega \\
\nabla\cdot\mathbf{u} & = & 0 & \text{in } \Omega \\
\mathbf{u}(0,y) = \mathbf{u}(2.2,y) & = & 0.41^{-2}\,(1.2y(0.41-y),0) & 0 \le y \le 0.41 \\
\mathbf{u}(x,0) = \mathbf{u}(x,0.41) & = & (0,0) & 0 \le x \le 2.2 \\
\mathbf{u}(S) & = & (0,0). &
\end{array} \tag{3.1}$$

Benchmark coefficients are the drag coefficient c_D, the lift coefficient c_L, and the

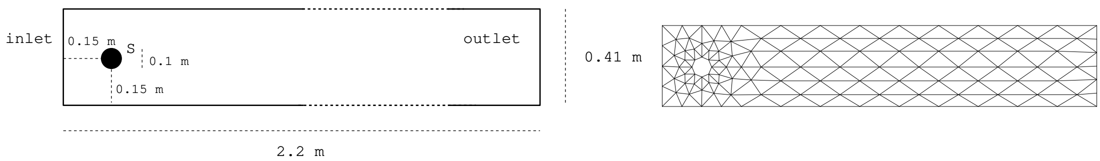

Figure 1: Domain and coarsest grid (level 0) for the numerical tests.

difference of the pressure between the front and the back of the cylinder Δp. The time dependent problem describes a Kármán vortex street by

$$\begin{array}{rcll}
\mathbf{u}_t - 10^{-3}\Delta\mathbf{u} + (\mathbf{u}\cdot\nabla)\mathbf{u} + \nabla p & = & \mathbf{0} & \text{in } \Omega\times[t_0, t_0+T] \\
\nabla\cdot\mathbf{u} & = & 0 & \text{in } \Omega\times[t_0, t_0+T] \\
\mathbf{u}(0,y) = \mathbf{u}(2.2,y) & = & \mathbf{g}(y) & 0 \le y \le 0.41, t_0 \le t \le T \\
\mathbf{u}(x,0) = \mathbf{u}(x,0.41) & = & (0,0) & 0 \le x \le 2.2, t_0 \le t \le T \\
\mathbf{u}(S) & = & (0,0) & t_0 \le t \le T \\
\mathbf{u}(t_0) & = & \mathbf{u}_0 &
\end{array} \tag{3.2}$$

with $\mathbf{g}(y) = 0.41^{-2}\,(6y(0.41-y),0)$. The benchmark coefficients are $c_D(t)$, $c_L(t)$ and $\Delta p(t)$. The time t_0 corresponds to a state where c_L has a maximum. The length of a period is denoted by T. Additional benchmark coefficients are the maximal values of $c_D(t)$ and $c_L(t)$, $\Delta p(t_0 + T/2)$, and the Strouhal number St.

We have used the coarsest grid (level 0) depicted in Figure 1. The data of the grids which results from uniform refinement can be seen in Table 3.1. The benchmark coefficients for (3.1) which have been computed with the nonconforming P_1/P_0–finite element discretization are presented in Table 1 and benchmark coefficients for (3.2) in Table 2.

level	velocity	pressure	edges on S	c_D	c_L	Δp
0	532	192	8			
3	36512	12288	64	5.6797	5.040-3	0.1135
4	146752	49152	128	5.6132	9.745-3	0.1154
5	588416	196608	256	5.5889	1.049-2	0.1166
			benchmark interval	[5.5700, 5.5900]	[1.040-2, 1.100-2]	[0.1172 0.1176]

Table 1: Example (3.1), number of unknowns and computed benchmark coefficients.

level	τ	$c_{D_{max}}$	$c_{L_{max}}$	$\Delta p(t_0 + T/2)$	St
5	0.01	3.2419	0.9689	2.4673	0.2941
5	0.005	3.2377	0.9647	2.4615	0.2985
5	0.0025	3.2370	0.9624	2.4610	0.2963
benchmark interval		[3.2200, 3.2400]	[0.9800, 1.020]	[2.4600, 2.5000]	[0.2950, 0.3050]

Table 2: Example (3.2), computed benchmark coefficients.

4 The Numerical Tests

The following investigations have been subject to our numerical tests:

1. a comparison of the V–, F–, and W–cycle of the coupled multigrid method,
2. the influence of the number of smoothing steps on the computing times in the W–cycle,
3. the Vanka smoother and stabilized Vanka smoother in the W–cycle,
4. the significance of the step length control,
5. the different approaches of the coarsest grid iteration for the W–cycle,
6. a study of the parallel scaling of the V–, F–, and W–cycle.

The termination criterion of the nonlinear iteration has been the Euclidean norm of the residual to be lower than 10^{-7} in Example (3.1) and 10^{-5} in Example (3.2). If not stated otherwise, the tests have been carried out using the step length control, the standard approach of the coarsest grid iteration with a reduction factor of 0.1, and the interpolated solution of the next coarser grid as starting point of the nonlinear iteration. The number of pre– and post–smoothing steps has been the same on all grids except the coarsest grid. The smoothing steps have not been damped.

Next we explain the abbreviations in the tables of this section:

Vanka–V(F,W)	coupled multigrid method with V–cycle, (F–cycle, W–cycle), Vanka smoother,
stab. Vanka	coupled multigrid method with W–cycle, stabilized Vanka smoother,
M	maximal number of multigrid cycles per nonlinear iteration step,
sm.	number of smoothing steps,
lev.	level for the computations,
pr.	number of processors = 2 CPU with one local memory,
it.	number of nonlinear iterations/number of multigrid cycles,
time	cpu time in seconds,
comm.	part of the communication on the cpu time in %,
lev0.	part of the iteration on the coarsest grid on the cpu time in %,
time/cyc.	time for one multigrid cycle in seconds,
τ	fixed length of the time step in Example (3.2),
t/per.	computing time for one period in Example (3.2),
t/step	average computing time for one time step in Example (3.2).

The computations on one processor have been carried out, if not stated otherwise, on a workstation HP 9000 J 210 (120 MHz, 22.1 MFlops (LINPACK)). As parallel computer we could use a Parsytec GCPowerPlus (80 MHz, 9.2 MFlops/processor (LINPACK)) which is known of having a better computing performance than communication performance. The implementation of the algorithms has been done using the program package *ugp*1.0 by Bastian [Bas]. In particular data structures, load balancing routines and the parallel overhead of this program have been used. However, the data structures had to be extended in order to handle multidimensional data and nonconforming finite elements.

1. A comparison of coupled multigrid V–, F–, and W–cycle.

With the help of coupled multigrid methods an approximation of the solution of the linear system (2.1) is computed. In a first test series a relatively small amount of numerical work has been invested in that approximation, see the upper part of Table 3. The lower part of Table 3 presents results of tests where a large amount of numerical work is used in the solution of the linear systems. With that the number of nonlinear iteration steps decreases considerably which leads to shorter computing times, too. The number of cycles M and of smoothing steps have been chosen such that the numerical work of the V–, F–, and W–cycle invested in the solution of (2.1) was similar within each test series.

The results for the steady state problem, Table 3, show the supremacy of the W–cycle for the solution of Example (3.1) on one processor. The superiority decreases with an increase of the number of processors. For 8 and 16 processor the computing times of the F– and W–cycle are similar. The V–cycle also becomes better in comparison to the W–cycle but still is slowest. The improvement of the V– and F–cycle is a result of their better scaling on parallel computers. The W–cycle spends an enormous amount of computing time on the coarsest grid and needs much more communication time than the V– and F–cycle. If 8 processors are used for computing the problem on level 4 then the load of each processor is nearly 50 %. In these computations the

solver	M	sm.	lev.	pr.	it.	time	comm.	lev0.
Vanka–V	5	5	4	1	55/271	15400	0.0	1.0
Vanka–F	5	4	4	1	24/118	7280	0.0	4.4
Vanka–W	5	3	4	1	18/88	4770	0.0	8.6
Vanka–V	5	5	4	8	57/279	2790	4.5	7.1
Vanka–F	5	4	4	8	31/153	2000	10.3	26.5
Vanka–W	5	3	4	8	29/141	2000	13.6	40.1
Vanka–V	5	5	5	16	45/222	4240	3.0	2.8
Vanka–F	5	4	5	16	31/151	3600	7.8	15.5
Vanka–W	5	3	5	16	24/166	3520	15.4	40.0
Vanka–V	10	8	4	1	19/181	15200	0.0	0.7
Vanka–F	10	6	4	1	11/96	8140	0.0	3.1
Vanka–W	10	5	4	1	9/73	5750	0.0	5.8
Vanka–V	10	8	4	8	21/200	2840	3.6	4.7
Vanka–F	10	6	4	8	12/105	1720	7.7	16.6
Vanka–W	10	5	4	8	11/97	1790	11.9	31.2
Vanka–V	10	8	5	16	22/208	5770	2.7	2.2
Vanka–F	10	6	5	16	12/106	3330	6.5	10.8
Vanka–W	10	5	5	16	9/82	3240	13.1	31.1

Table 3: Example (3.1), comparison of the coupled multigrid V–, F–, and W–cycle.

communication overhead of the V– and F–cycle is larger than on 16 processors which are loaded well (level 5).

In the time dependent problem the alternative approach of the coarsest grid iteration with 20 steps has been used for the coupled multigrid F– and W–cycle. The

solver	M	sm.	lev.	τ	pr.	it.	t/per.	t/step	comm.	lev0.
Vanka–V	5	5	4	0.01	8	8	14100	382	4-6	3-6
Vanka–F	5	4	4	0.01	8	6	10400	281	4.4	3.7
Vanka–W	5	3	4	0.01	8	5-6	9010	244	6.3	8.2
Vanka–V	5	5	4	0.005	8	4-5	14700	204	4-6	2-6
Vanka–F	5	4	4	0.005	8	3-4	11600	161	4.6	3.7
Vanka–W	5	3	4	0.005	8	3-4	10500	146	6.4	8.2
Vanka–V	5	5	4	0.0025	8	3	16700	117	5.0	3-5
Vanka–F	5	4	4	0.0025	8	3	16400	114	4.4	3.6
Vanka–W	5	3	4	0.0025	8	3	14600	102	5.9	8.1
Vanka–V	5	5	5	0.01	16	12-17	48900	1359	2.9	1.1
Vanka–F	5	4	5	0.01	16	9-12	37600	1045	3.6	2.7
Vanka–W	5	3	5	0.01	16	9-10	32300	896	6.7	9.2
Vanka–V	5	5	5	0.005	16	8-10	56500	807	2.4	1.1
Vanka–F	5	4	5	0.005	16	6-7	42300	605	3.6	2.7
Vanka–W	5	3	5	0.005	16	6	38000	543	7.1	9.0
Vanka–V	5	5	5	0.0025	16	4-5	56800	411	2.9	1.1
Vanka–F	5	4	5	0.0025	16	3-4	43100	312	3.6	2.7
Vanka–W	5	3	5	0.0025	16	3	39400	285	6.6	9.1

Table 4: Example (3.2), comparison of the coupled multigrid V–, F–, and W–cycle.

results of Table 4 show that the W–cycle performs best for all time steps τ. Often, the F–cycle is somewhat slower and with a different choice of parameters the situation may be vice versa. Only the V–cycle needs considerably more computing time, especially for large time steps.

2. The number of smoothing steps in the W–cycle.

The number of smoothing steps is an important parameter in multigrid methods. A good choice may speed up a computation considerably.

solver	M	sm.	lev.	pr.	it.	time	comm.	lev0.
Vanka–W	10	2	5	1	13/121	19100	0.0	5.6
Vanka–W	10	3	5	1	10/88	18700	0.0	4.2
Vanka–W	10	5	5	1	7/61	19400	0.0	2.7
Vanka–W	10	10	5	1	5/35	20400	0.0	1.4
Vanka–W	10	2	5	16	17/166	4180	17.1	47.7
Vanka–W	10	3	5	16	14/129	3860	15.3	40.3
Vanka–W	10	5	5	16	9/82	3240	13.1	31.1
Vanka–W	10	10	5	16	6/49	3100	10.4	20.0

Table 5: Example (3.1), the coupled multigrid W–cycle with different numbers of smoothing steps.

Results of a numerical study of the influence of the number of smoothing steps on computing times are presented in Table 5. The main difference of the two test series has been the number of processors. It can be observed that an increase of the number of smoothing steps results in a decrease of the number of nonlinear iterations and multigrid cycles. But on a serial computer this does not pay off. On the contrary, the computing times increase with an increase of the number of smoothing steps. However, the situation changes completely on 16 processors. Now the computing times are reduced with an increase of the number of smoothing steps. The reason for that can be found in the ratio of the numerical work to parallel work in a smoothing step. This ratio is the better the finer the grid. With an increase of the number of smoothing steps the influence of the numerical work on finer grids increases whereas the importance of the communications in the coarsest grid iteration diminishes.

This is an example where the good choice of an important parameter of an algorithm substantially depends on the number of processors.

3. The Vanka smoother and stabilized Vanka smoother in the W–cycle.

Table 6 presents a comparison of the Vanka smoother and the stabilized Vanka smoother for Example (3.1). The stabilized Vanka smoother shows the same properties as the Vanka smoother concerning the computing times on different numbers of processors. For one processor a small number of smoothing steps is best and for 8 and 16 processors a large number. The stabilized Vanka smoother beats the Vanka smoother in all test series. The superiority on 8 and 16 processors comes also from the better ratio of numerical to parallel work in the stabilized Vanka smoother. The number and length of communications coincide, but the number of floating point operations using the stabilized Vanka smoother is three times larger.

smoother	sm.	lev.	pr.	it.	time	comm.	lev0.
Vanka–W	3	4	1	10/89	4800	0.0	9.8
stab. Vanka	1	4	1	11/102	3290	0.0	15.7
stab. Vanka	2	4	1	9/82	4000	0.0	9.9
stab. Vanka	4	4	1	7/68	4810	0.0	6.1
Vanka–W	3	4	8	14/127	1750	13.3	39.8
stab. Vanka	1	4	8	17/166	1420	11.4	40.3
stab. Vanka	2	4	8	12/111	1330	9.3	29.9
stab. Vanka	4	4	8	8/65	1210	7.3	19.7
Vanka–W	3	5	16	14/129	3860	15.3	40.3
stab. Vanka	1	5	16	18/175	3250	12.6	41.7
stab. Vanka	2	5	16	11/101	2630	10.4	31.6
stab. Vanka	4	5	16	6/50	2000	8.2	21.0

Table 6: Example (3.1), comparison of different smoothers, $\mathsf{M} = 10$

4. The significance of the step length control.

Here we want to study if the additional work which comes from the step length control (2.2) pays off.

Results for the steady state case are presented in Table 7. They show the importance of the step length control for the F– and W–cycle. The computing times are reduced considerably in some tests. Only the V–cycle does not benefit from the step length control. On level 4 the computing time increases due to the additional work for computing κ and on level 5 even the number of nonlinear iterations increases. This shows that the step length control makes sense only if the underlying iterative method computes good updates.

solver	slc.	M	sm.	lev.	pr.	it.	time	time/cyc.
Vanka–V	no	10	8	4	4	21/205	5440	26.5
Vanka–V	yes	10	8	4	4	21/198	5550	28.0
Vanka–F	no	10	4	4	4	18/174	3770	21.7
Vanka–F	yes	10	4	4	4	17/158	3570	22.6
Vanka–W	no	10	3	4	4	17/161	3740	23.2
Vanka–W	yes	10	3	4	4	13/119	2790	23.4
Vanka–V	no	10	8	5	16	17/164	4370	26.6
Vanka–V	yes	10	8	5	16	22/208	5770	27.7
Vanka–F	no	10	4	5	16	20/192	4350	22.7
Vanka–F	yes	10	4	5	16	15/139	3240	23.2
Vanka–W	no	10	3	5	16	16/150	4200	28.0
Vanka–W	yes	10	3	5	16	14/129	3860	29.9

Table 7: Example (3.1), the step length control (slc) in coupled multigrid methods.

The influence of the step length control in the time dependent problem (3.2) can be seen in comparing the results of Table 8 and Table 4. The application of the step length control reduces the computing times for all methods substantially, between 30% and 50%.

solver	M	sm.	lev.	τ	pr.	it.	t/per.	t/step	comm.	lev0.
Vanka–V	5	5	4	0.005	8	9	27900	387	3.0	1.8
Vanka–F	5	4	4	0.005	8	6-7	20100	280	4.1	3.9
Vanka–W	5	3	4	0.005	8	6	18200	252	6.1	8.7
Vanka–V	5	5	5	0.005	16	14-17	94500	1349	2.4	1.2
Vanka–F	5	4	5	0.005	16	10-11	69700	996	3.7	2.8
Vanka–W	5	3	5	0.005	16	9-10	61200	874	7.0	9.5

Table 8: Example (3.2), coupled multigrid methods without step length control.

5. The coarsest grid iteration in the W–cycle.

We consider the coupled multigrid W–cycle because of the enormous percentage of computing time on the coarsest grid. The standard approach with an reduction factor of 0.1 has been compared with the alternative approach having a fixed number of iterations – 10, 20, and 30. The starting point of the nonlinear iteration was $(\mathbf{0}, 0)$.

The results in Table 9 show the expected behaviour of the method. The shortest time on one processor is needed with the standard approach and the alternative approach with a large number of steps. On 4 processors the alternative approach with 20 and 30 steps is faster than the standard approach. On 16 processors the standard approach takes the longest computing time. The tendency is clear. The

solver	coarsest grid it.	M	sm.	lev.	pr.	it.	time	comm.	lev0.
Vanka–W	stand., red. 0.1	10	3	4	1	19/176	9050	0.0	6.1
Vanka–W	altern., 10 steps	10	3	4	1	26/247	12000	0.0	0.7
Vanka–W	altern., 20 steps	10	3	4	1	21/203	9900	0.0	1.3
Vanka–W	altern., 30 steps	10	3	4	1	19/184	9050	0.0	1.9
Vanka–W	stand., red. 0.1	10	3	4	4	23/216	5090	10.0	33.0
Vanka–W	altern., 10 steps	10	3	4	4	35/341	5650	3.9	4.5
Vanka–W	altern., 20 steps	10	3	4	4	27/260	4500	4.1	8.5
Vanka–W	altern., 30 steps	10	3	4	4	25/235	4240	4.4	12.0
Vanka–W	stand., red. 0.1	10	3	5	16	50/489	14900	15.5	41.9
Vanka–W	altern., 10 steps	10	3	5	16	68/673	13000	6.8	9.0
Vanka–W	altern., 20 steps	10	3	5	16	63/621	12900	7.3	16.1
Vanka–W	altern., 30 steps	10	3	5	16	60/590	13300	6.9	22.3

Table 9: Example (3.1), coupled multigrid W–cycle with different variants of the coarsest grid iteration.

more processors are used the less work should be invested on the coarsest grid. The increasing number of multigrid cycles is more than compensated with the savings on the coarsest grid.

6. The parallel scaling of the V–, F–, and W–cycle.

For studying the scaling of the coupled multigrid methods we have started on one processor with Example (3.1) and level 3. Then the number of processors as well as the size of the problem have been increased by a factor of 4. Thus, for all tests the processors have been loaded well. The parallel scaling is investigated considering the computing times for a fixed number of multigrid cycles. The results of Table 10 were obtained with 100 multigrid cycles.

solver	M	sm.	lev.	pr.	it.	time	comm.	lev0.
Vanka–V	10	3	3	1	10	1290	0.0	11.0
Vanka–V	10	3	4	4	10	1273	3.1	7.2
Vanka–V	10	3	5	16	10	1267	3.6	4.0
Vanka–F	10	3	3	1	10	1810	0.0	23.8
Vanka–F	10	3	4	4	10	1860	7.2	20.8
Vanka–F	10	3	5	16	10	1880	8.4	16.6
Vanka–W	10	3	3	1	10	1990	0.0	28.6
Vanka–W	10	3	4	4	10	2330	10.2	32.2
Vanka–W	10	3	5	16	10	2910	14.9	39.2

Table 10: Example (3.1), scaling of the coupled multigrid methods.

First it can be observed that the computing times for the coupled multigrid V–cycle decrease. The increase of the level has led to a reduction of the time on the coarsest grid which has compensated the communications on 4 and 16 processors. The nearly constant times for one V–cycle can be seen also in Table 7. These results demonstrate the very good performance of the nonconforming P_1/P_0–finite element which comes from the absence of crosspoints. The F–cycle shows a similar behaviour as the V–cycle. The losses due to the communication overhead are small, compare also Table 7. Only the W–cycle shows considerable losses of efficiency. Increasing the number of processors as well as the size of the problem by a factor of 4 the execution time per cycle increases by a factor of 1.2 to 1.3. The reasons for that have been discussed already.

5 Summary of the Numerical Results

We want to give a short summary of the study concerning the parallel behaviour of coupled multigrid methods for incompressible Navier–Stokes equations.

1. The coupled multigrid W–cycle has performed best in most tests as well for the steady state as for the time dependent equations. This supremacy was greatest on one processor. On 4 and more processors the F–cycle needed similar computing times. The V–cycle was always the slowest method. However, with a larger number of processors the difference to the W–cycle has diminished.
2. For a small number of processors one should choose a small number of smoothing steps in the coupled multigrid W–cycle. For a large number of processors a large number of smoothing steps has resulted in shorter computing times.
3. The solution of the problems was faster using the stabilized Vanka smoother instead of the Vanka smoother.
4. The step length control has improved the convergence of the coupled multigrid methods in most computations substantially.

5. For a small number of processors one should use the standard approach of the coarsest grid iteration in the coupled multigrid W–cycle. For a large number of processors the alternative approach with a small number of steps was more efficient.
6. The coupled multigrid V– and F–cycle have shown a very good parallel scaling because of the absence of crosspoints in the nonconforming P_1/P_0-finite element discretization.

References

[AN] O. Axelsson and M. Neytcheva. Scalable algorithms for the solution of Navier's equations of elasticity. *J. Comput. Appl. Math.*, 63:149 – 178, 1995.

[Bas] P. Bastian. *Parallele adaptive Mehrgitterverfahren.* Teubner Skripten zur Numerik. Teubner–Verlag, 1996.

[CR] M. Crouzeix and P.-A. Raviart. Conforming and nonconforming finite element methods for solving the stationary Stokes equations I. *R.A.I.R.O. Analyse Numérique*, 7:33–76, 1973.

[Joh] V. John. *Parallele Lösung der inkompressiblen Navier–Stokes Gleichungen auf adaptiv verfeinerten Gittern.* PhD thesis, Otto-von-Guericke Universität Magdeburg, Fakultät für Mathematik, 1997. to appear.

[Sch1] F. Schieweck. A parallel multigrid algorithm for solving the Navier–Stokes equations. *Impact Comp. Sci. Engrg.*, 5:345 – 378, 1993.

[Sch2] F. Schieweck. *Parallele Lösung der stationären inkompressiblen Navier–Stokes Gleichungen.* Otto-von-Guericke Universität Magdeburg, Fakultät für Mathematik, 1996. Habilitation.

[ST1] M. Schäfer and S. Turek. The benchmark problem "Flow around a cylinder". In E.H. Hirschel, editor, *Flow Simulation with High–Performance Computers II*, volume 52 of *Notes on Numerical Fluid Mechanics.* Vieweg, 1996.

[ST2] F. Schieweck and L. Tobiska. An optimal order error estimate for an upwind discretization of the Navier–Stokes equations. *Num. Meth. Part. Diff. Equ.*, 12:407 – 421, 1996.

[Tur] St. Turek. A comparative study of time–stepping techniques for the incompressible Navier–Stokes equations: From fully implicit non–linear schemes to semi–implicit projection methods. *Int. J. Num. Meth. Fluids*, 22:987 – 1011, 1996.