INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS, VOL. 26, 281–301 (1998)

ALGEBRAIC MULTIGRID METHODS FOR THE SOLUTION OF THE NAVIER–STOKES EQUATIONS IN COMPLICATED GEOMETRIES

MICHAEL GRIEBEL,[1]* TILMAN NEUNHOEFFER[2] AND HANS REGLER[2]

[1] *Institut für Angewandte Mathematik, Abteilung Wissenschaftliches Rechnen und Numerische Simulation, Universität Bonn, Wegelerstraße 6, D-53115 Bonn, Germany;*
[2] *Institut für Informatik, Technische Universität München, D-80290 München, Germany*

SUMMARY

The application of standard multigrid methods for the solution of the Navier–Stokes equations in complicated domains causes problems in two ways. First, coarsening is not possible to full extent since the geometry must be resolved by the coarsest grid used. Second, for semi-implicit time-stepping schemes, robustness of the convergence rates is usually not obtained for convection–diffusion problems, especially for higher Reynolds numbers. We show that both problems can be overcome by the use of algebraic multigrid (AMG), which we apply for the solution of the pressure and momentum equations in explicit and semi-implicit time-stepping schemes. We consider the convergence rates of AMG for several model problems and demonstrate the robustness of the proposed scheme. © 1998 John Wiley & Sons, Ltd.

Int. J. Numer. Meth. Fluids, **26**: 281–301 (1998).

KEY WORDS: Navier–Stokes equations; SIMPLE algorithm; algebraic multigrid methods

1. INTRODUCTION

In this paper we consider a fast solver for the numerical simulation of two- and three-dimensional viscous, non-stationary, incompressible fluid flow problems in complicated geometries such as arise in the study of porous medium flow on a microscale level, in multi-connected technical devices such as cooling or heating systems or in a vast number of biological and medical flow simulations (see Figure 1). Also for free boundary problems where the domain changes in time, partial differential equations in rather complicated geometries have to be solved in each time step.

If we use an explicit time discretization such as the forward Euler scheme, most of the computational effort has to be spent in the solution of the Poisson equation in a pressure correction step. Semi-implicit discretization schemes such as the backward Euler scheme allow larger time steps, but here, besides the Poisson equation, we additionally obtain convection–diffusion equations for each component of the velocity.

Multigrid methods are often-used solvers for the arising algebraic equations. However, for convection-dominated convection–diffusion problems such as appear for high Reynolds numbers, standard multigrid methods show a lack of robustness with respect to the convergence behaviour.

Correspondence to: M. Griebel, Inst. für Angewandte Math., Universitat Bonn, Wegelerstraße 6, D-53115 Bonn, Germany.

CCC 0271–2091/98/030281–21 $17.50
Received October 1996
Revised March 1997

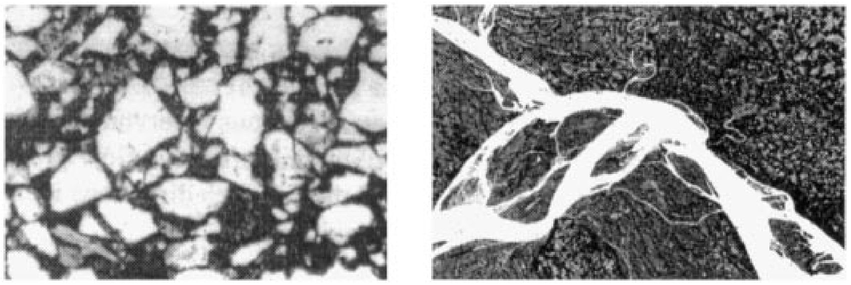
Figure 1. Examples of flow problems in fixed complicated geometries: left, a river system; right, a porous medium (cross-section of sandstone)[1]

This can be overcome by some special techniques for the construction of the restriction and coarse grid operators and the use of ILU smoothers.[2–6] This only works well in two spatial dimensions. The second drawback is that the geometry of the domain must be resolved on the coarsest level of discretization used in the multigrid method. Thus the domain is not allowed to have a complicated structure, otherwise there would be many unknowns on the coarsest level and an iterative scheme would need too many smoothing steps on the coarsest grid to maintain good convergence rates, whereas direct solvers for the coarse grid equation are too expensive. Furthermore, it is not possible to use grid transformation techniques to transform a non-rectangular physical domain into a rectangular computational domain if the domain is too complicated. For some other concepts on the application of multigrid methods to problems on complex domains, see References 7–10.

In the 1980s, algebraic multigrid methods were introduced. They do not make use of any geometric information on the grid. Here the coarse grid points and the restriction and interpolation operators are constructed by considering only the linear system and the coupling between the different unknowns. In numerical experiments, algebraic multigrid has been shown to possess advantages over conventional multigrid methods with respect to *robustness*.[11,12] Their convergence rates are bounded by a constant $C < 1$ independent of the PDE under consideration, also for problems with strongly varying coefficient functions or singular perturbed problems, e.g. diffusion problems or convection–diffusion problems with strong anisotropy or strong convection respectively. Robust convergence rates are also obtained for problems on domains with *complicated geometry*, even for the additive variant of AMG.[13]

We apply AMG to the equations arising from explicit and semi-implicit time discretizations of the Navier–Stokes equations and present the results of numerical experiments where we study the dependence of the convergence rates on the geometry, the Reynolds number and the number of unknowns. For other concepts on the application of AMG to the Navier–Stokes equations, see References 14–16.

2. DISCRETIZATION OF NAVIER–STOKES EQUATIONS

2.1. Navier–Stokes Equations

We consider the time-dependent, incompressible Navier–Stokes equations for the velocity $\vec{u}$ and the kinematic pressure p which is defined as the real pressure divided by the density, in an arbitrary bounded domain $\Omega \subset \mathbb{R}^2$ or $\mathbb{R}^3$:

$$\vec{u}_t - \frac{1}{Re}\Delta\vec{u} + \vec{u}\cdot\nabla\vec{u} + \nabla p = \vec{g}, \tag{1}$$

$$\nabla\cdot\vec{u} = 0. \tag{2}$$

Equation (1) is the momentum equation and equation (2) is the continuity equation. *Re* denotes the Reynolds number and $\vec{g}$ is the body force, e.g. gravity. In addition, we need suitable initial conditions $\vec{u}|_{t=0} = \vec{u}_0$ and boundary conditions of inflow, outflow, slip or no-slip type. This means that either the velocity itself ($\vec{u}|_\Gamma = \vec{u}_\Gamma$) or its normal derivative $((\partial\vec{u}/\partial n)|_\Gamma = (\partial\vec{u}/\partial n)_\Gamma$, where $\vec{n}$ denotes the unit outer normal vector at the boundary Γ) is specified at the boundary. For a detailed description of the different boundary types, see e.g. References 17 and 18 (pp. 12ff).

The initial condition must satisfy $\nabla \cdot \vec{u}_0 = 0$ and $\vec{u}_0|_\Gamma \cdot \vec{n} = \vec{u}_\Gamma|_{t=0} \cdot \vec{n}$[19]. An initial velocity field $\vec{u}_0$ satisfying $\nabla \cdot \vec{u}_0 = 0$ can be obtained by solving the potential equation

$$\Delta\Phi = 0, \qquad \nabla\Phi|_\Gamma \cdot \vec{n} = \vec{u}_\Gamma|_{t=0} \cdot \vec{n} \tag{3}$$

and setting

$$\vec{u}_0 := \nabla\Phi. \tag{4}$$

2.2. *Discretization in space*

For space discretization we use finite differences on a staggered grid with equidistant orthogonal grid lines, which was first introduced by Harlow and Welch.[20] The convective parts are discretized by flux blending, i.e. a mixture of central and upwind differences, namely the donor–cell scheme, such that the discretization does not suffer from instabilities.[17]* If Ω is non-rectangular, we approximate Ω by a domain Ω_h such that the boundary of Ω_h coincides with grid lines. Then we imbed Ω_h in a rectangular domain $\tilde{\Omega} \supset \Omega_h$. Thus $\tilde{\Omega}$ can be divided into the set of fluid cells representing Ω_h and a set of boundary cells (see Figure 2).

Details of the discretization in space and of the discretization of the boundary conditions can be found in References 17 and 18 (Chap. 3).

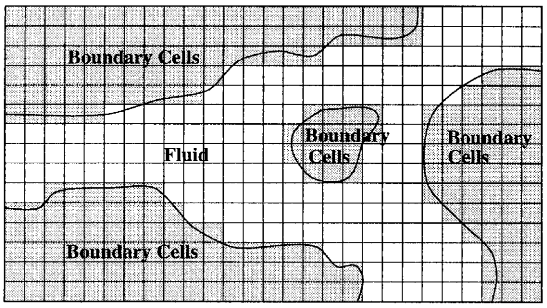

Figure 2. Imbedding of a non-rectangular domain

* There exist several other stable discretization schemes for convection–diffusion problems, e.g. streamline and upwind diffusion methods[21,22] or Petrov–Galerkin methods.

2.3. *Discretization in time*

For time discretization we use either the forward or the backward Euler scheme.† The explicit forward Euler scheme with time step $\delta t := t_{n+1} - t_n$ leads to the following coupled problem: find $\vec{u}^{(n+1)}$, $p^{(n+1)}$ such that

$$\vec{u}^{(n+1)} + \delta t \nabla p^{(n+1)} = \vec{u}^{(n)} + \delta t \left(\frac{1}{Re} \Delta \vec{u}^{(n)} - \vec{u}^{(n)} \cdot \nabla \vec{u}^{(n)} + \vec{g}^{(n)} \right), \tag{5a}$$

$$\nabla \cdot \vec{u}^{(n+1)} = 0, \tag{5b}$$

where the index (n) denotes velocity and pressure at time t_n.

For its solution we first choose a tentative velocity field

$$\vec{u}^* := \vec{u}^{(n)} + \delta t \left(\frac{1}{Re} \Delta \vec{u}^{(n)} - \vec{u}^{(n)} \cdot \nabla \vec{u}^{(n)} + \vec{g}^{(n)} \right) \tag{6}$$

and then obtain $\vec{u}^{(n+1)}$ by adding the gradient of the pressure in the new time step:

$$\vec{u}^{(n+1)} = \vec{u}^* - \delta t \nabla p^{(n+1)}. \tag{7}$$

Replacing $\vec{u}^{(n+1)}$ by the right-hand side of (7) in the continuity equation (5b) leads to a Poisson equation for the pressure:

$$\Delta p^{(n+1)} = \frac{1}{\delta t} \nabla \cdot \vec{u}^*. \tag{8}$$

Thus we first solve (8) and then compute $\vec{u}^{(n+1)}$ by means of (7). Suitable boundary conditions are homogeneous Neumann conditions $\partial p/n = 0$ for the pressure and $\vec{u}^* = \vec{u}^{(n+1)}$ for the tentative velocity field.

For reasons of stability of the time-stepping scheme, we must observe some restrictive conditions on the step size δt, the so-called Courant–Friedrichs–Lewy conditions, which guarantee that no particle of the fluid can pass more than one grid line in any direction in one time step, i.e.

$$\max_{x \in \Omega} |u_i(x)| \delta t < \delta x_i, \quad i = 1, 2, (3). \tag{9}$$

Larger time steps are possible if we use an implicit time-stepping scheme such as the backward Euler scheme. Here we apply a semi-implicit discretization of the convection term to avoid non-linearities in the algebraic equations. Thus we end up with the following coupled problem: find $\vec{u}^{(n+1)}$, $p^{(n+1)}$ such that

$$\vec{u}^{(n+1)} + \delta t \left(-\frac{1}{Re} \Delta \vec{u}^{(n+1)} + \vec{u}^{(n)} \cdot \nabla \vec{u}^{(n+1)} + \nabla p^{(n+1)} \right) = \vec{u}^{(n)} + \delta t \vec{g}^{(n+1)}, \tag{10a}$$

$$\nabla \cdot \vec{u}^{(n+1)} = 0. \tag{10b}$$

† There also exist more elaborate time-stepping schemes, even of higher order, e.g. the Crank–Nicolson scheme or the fractional step θ-scheme.[23] Note that also in those schemes, basically Poisson equations and convection–diffusion equations have to be solved.

Again we compute a tentative velocity field, now as the solution of the convection–diffusion equation*

$$\vec{u}^* + \delta t\left(-\frac{1}{Re}\Delta\vec{u}^* + \vec{u}^{(n)}\cdot\nabla\vec{u}^* + \nabla p^{(n)}\right) = \vec{u}^{(n)} + \delta t\vec{g}^{(n+1)}. \tag{11}$$

These velocities do not satisfy the continuity equation. Thus we have to compute some correction terms $\vec{u}'$ and p' such that

$$\vec{u}^{(n+1)} = \vec{u}^* + \vec{u}', \qquad p^{(n+1)} = p^{(n)} + p'. \tag{12}$$

Subtracting (11) from (10a) gives us an equation for $\vec{u}'$, i.e.

$$\vec{u}' + \delta t\left(-\frac{1}{Re}\Delta\vec{u}' + \vec{u}^{(n)}\cdot\nabla u' + \nabla p'\right) = 0, \tag{13}$$

or, using the abbreviation

$$S(\vec{u}') := -\frac{1}{Re}\Delta\vec{u}' + \vec{u}^{(n)}\cdot\nabla\vec{u}', \tag{14}$$

we obtain

$$\vec{u}' + \delta tS(\vec{u}') + \delta t\nabla p' = 0. \tag{15}$$

Depending on the specific choice of how to approximate $S(\vec{u}')$, there exist different numerical methods in the literature.

One is to approximate $S(\vec{u}')$ by zero. This implies that the values of $\vec{u}'$ change very little in space. Then we put (15) into the continuity equation and obtain a Poisson equation for the pressure correction:

$$\Delta p' = \frac{1}{\delta t}\nabla\cdot\vec{u}^*. \tag{16}$$

Again, homogeneous Neumann boundary conditions are used for the pressure correction as for the pressure in the scheme above.

This approach is used for example in References 24 and 25, where this scheme is called SIMPLE, as well as in Reference 26, where this scheme is derived from the SMAC (simplified marker and cell) method.[27] In the following we call this scheme SMAC.

In the original SIMPLE scheme introduced by Patankar and Spalding,[28] the space discretization of the linear operator $S(\vec{u}')$ is written as the product of a matrix A, which depends on the velocities in the previous time step, and the velocity correction $\vec{u}'_h$, i.e.

$$S_h(\vec{u}'_h) = A(\vec{u}_h^{(n)})\vec{u}'_h, \tag{17}$$

and (15) becomes

$$(I + \delta tA(\vec{u}_h^{(n)}))\vec{u}'_h + \delta t\nabla_h p'_h = 0. \tag{18}$$

Furthermore, $A(\vec{u}_h^{(n)})$ is replaced by the diagonal matrix $D(\vec{u}_h^{(n)}) := \text{diag}(A(\vec{u}_h^{(n)}))$. Thus $\vec{u}'_h$ is assumed to be small and the off-diagonal elements are neglected. Equation (18) gives

$$\vec{u}'_h = -\delta t(I + \delta tD(\vec{u}_h^{(n)}))^{-1}\nabla_h p'_h. \tag{19}$$

* Note that this is a system of decoupled convection–diffusion equations for each component of the tentative velocity field $\vec{u}^*$.

If we replace $\vec{u}'_h$ by the right-hand side of (19) in the continuity equation, we end up with an equation for p'_h which depends on $\vec{u}_h^{(n)}$:

$$\nabla_h \cdot (I + \delta t D(\vec{u}_h^{(n)}))^{-1} \nabla_h p'_h = \frac{1}{\delta t} \nabla \cdot \vec{u}_h^*. \tag{20}$$

Here relaxation parameters are often used for stationary problems to get better convergence results.

The SIMPLEC algorithm[29] assumes that $\vec{u}'_h$ is nearly constant in a certain surrounding and uses lumping of the matrix $I + \delta t A(\vec{u}_h^{(n)})$ in (18). This means that we replace the matrix $I + \delta t A(\vec{u}_h^{(n)})$ by a diagonal matrix where the diagonal elements are the sums of all elements of one row.*

Moreover, there exist schemes such as SIMPLER[30] which use an inner iteration of at most two cycles for the solution of the coupled problem (10a,b). One cycle consists of the computation of the tentative velocity field by (11) using the pressure compound in the previous cycle and the computation of a pressure correction by the continuity equation.

Of course, this is only a small selection of schemes to handle the pressure–velocity coupling. We want also to mention the projection scheme of Chorin[31] and Temam,[32] the PISO method of Issa[33] and the work done by van Kan,[34] among others. For a comparison of some schemes, see Reference 35.

For all these different numerical schemes we basically have to solve Poisson or Poisson-like equations for the pressure or the pressure correction and convection–diffusion equations for each component of the velocity vector. In the following we will consider the solution of these equations by algebraic multigrid.

2.4. *Transport equation*

Besides the Navier–Stokes equations, we also consider the scalar transport equation

$$c_t - \lambda \Delta c + \vec{u} \cdot \nabla c = 0, \tag{21}$$

with the diffusion coefficient λ and the velocity field $\vec{u}$.

This describes for example the transport of a chemical substance with concentration c. Setting $c := T$, (21) is the energy equation for the temperature T. Here, for reasons of simplicity, we omit the recoupling of the concentration or temperature respectively on the momentum equations, which can be modelled for example using the Boussinesq approximation.[36–38]

Time discretization with the backward Euler scheme gives

$$c^{(n+1)} + \delta t(-\lambda \Delta c^{(n+1)} + \vec{u} \cdot \nabla c^{(n+1)}) = c^{(n)}, \tag{22}$$

where $\vec{u}$ is either an already computed stationary velocity field or $\vec{u}^{(n+1)}$. Equation (22) is equivalent to the momentum equation (11) for the tentative velocity field $\vec{u}^*$. The only differences are the right-hand side and the diffusion coefficient. For the discretization in space we again use the staggered grid and finite differences with a mixed central/upwind discretization of the convective term. For details, see Reference 18 (pp. 134ff).

* Note that this approach is equivalent to the SMAC approach if we use conventional upwind discretizations instead of the donor–cell scheme.

3. ALGEBRAIC MULTIGRID METHOD AND ITS APPLICATION TO NAVIER–STOKES EQUATIONS

3.1. Algebraic multigrid

Algebraic multigrid methods for the solution of a linear system

$$\mathscr{A}_M u_M = f_M$$

on a fine grid level M were introduced in References 11, 12, 39 and 40. Here, first a grid is set up on the next coarser level by using algebraic information from $\mathscr{A}_L$ $(L \leqslant M)$ and then an appropriate interpolation scheme $\mathscr{P}^L_{L-1}$ is defined. After computing

$$\mathscr{A}_{L-1} := (\mathscr{P}^L_{L-1})^{\mathrm{T}} \mathscr{A}_L \mathscr{P}^L_{L-1} \tag{23}$$

via the Galerkin identity, the process is repeated until a sufficiently coarse level system is obtained. AMG is necessarily less efficient than highly specialized geometric multigrid solvers for elliptic problems on uniform rectangular grids. However, for more complicated cases with general domains, AMG has been shown to behave in a robust manner and thus performs quite favourably in terms of operation count and CPU time. AMG also works for problems where geometric multigrid methods are impossible to design. AMG uses no sophisticated smoother, but only standard Gauss–Seidel. The robustness of AMG is obviously the merit of the appropriately chosen grid-coarsening strategy and the associated interpolations.

For algebraic multigrid the grids should be nested as for conventional multigrid methods, but they need not be uniform. In fact, uniformity, if given for the finest grid, is in general not maintained in the process. We will nevertheless start with fine level discretizations based on the regular grid Ω_h. In the following we will denote the set of indices of the grid points corresponding to level L by N_L and we demand that the index sets be nested as

$$N_1 \subset N_2 \subset \cdots \subset N_{M-1} \subset N_M.$$

To each grid point of level L there corresponds an unknown of the solution vector u_L with the same index.

For an AMG algorithm the sequence of matrices $\mathscr{A}_L$ must be constructed algebraically. The $\mathscr{A}_{L-1}, L = M, \ldots, 2$, are computed successively by selecting a subset of the unknowns of the level L system by evaluating the *strength of the connections* between the unknowns in $\mathscr{A}_L$. The basis for our implementation is the AMG method described in References 11 and 12.

According to the well-known variational principle, it is best for a given interpolation to determine the coarse grid discretization via Galerkin coarsening. All error components lying in the range of the interpolation are then eliminated by a single coarse grid correction. In multigrid theory one has to take care that those error components, which are persistent to the smoother, are well represented on coarser grids.

The effect of Gauss–Seidel iterations on symmetric positive definite matrices $\mathscr{A}_M$ is well understood and can be used to guide the construction of the coarser level systems $\mathscr{A}_L$ for $L = M - 1, \ldots, 1$. Gauss–Seidel smoothing stalls whenever the error $e^{it}_L := u^{it}_L - u_L$ in iteration *it* is large in comparison with the residual $r^{it}_L := \mathscr{A}_L u^{it}_L - f_L$.

Because $\mathscr{A}_L e_L = r_L$, we have $\mathscr{A}_L e_L \approx 0$ then. Alternatively, for a single unknown,

$$(e_L)_i = -\frac{1}{(\mathscr{A}_L)_{ii}} \sum_{\substack{j=1 \\ j \neq i}}^{n_L} (\mathscr{A}_L)_{ij} (e_L)_j.$$

This sum may be split into the error components visible on the coarse grid (and thus eliminated by a single coarse grid correction step) and those which are not, i.e.

$$(e_L)_i = -\frac{1}{(\mathscr{A}_L)_{ii}}\left(\sum_{\substack{j\in C_L\\ j\neq i}}(\mathscr{A}_L)_{ij}(e_L)_j + \sum_{\substack{j\in F_L\\ j\neq i}}(\mathscr{A}_L)_{ij}(e_L)_j\right). \tag{24}$$

Here $C_L := N_{L-1}$ and $F_L := N_L\backslash N_{L-1}$. If the second sum could be eliminated on all levels, AMG would be a direct solver. In this case the ideal interpolation weights would be given by

$$(\mathscr{P}^L_{L-1}e_{L-1})_i = \begin{cases}(e_{L-1})_i, & i\in C_L,\\ -\dfrac{1}{(\mathscr{A}_L)_{ii}}\displaystyle\sum_{\substack{j\in C_L\\ j\neq i}}(\mathscr{A}_L)_{ij}(e_{L-1})_j, & i\in F_L.\end{cases} \tag{25}$$

Unfortunately, this ideal assumption can hardly be fulfilled when we want a decrease in the number of grid points on each level. Nevertheless, we try to minimize the second sum in (24) by choosing the coarse grid points $C_L := N_{L-1}$ from N_L appropriately.

We will briefly review the coarse grid selection part of AMG as introduced in References 11 and 12. For reasons of simplicity the level index L is omitted. Here we have to define the set of strongly coupled neighbours S_i of a point i. Let

$$d(i,I) := \frac{1}{\max\limits_{k\neq i}\{-\mathscr{A}_{ik}\}}\sum_{j\in I}-\mathscr{A}_{ij},$$

where I is any subset of N, and

$$S^i := \{j\in N|d(i,\{j\})\geqslant\alpha\},\qquad S^{i,\mathrm{T}} := \{j\in N|i\in S^j\}. \tag{26}$$

The partitioning into fine and coarse grid points is performed in two phases on each level. There we select coarse grid points in such a manner that as many strong couplings as possible are taken into consideration.

3.2. Selection of coarse grid points: set-up phase I

```
1. Set $C = \emptyset$ and set $F = \emptyset$
2. **While** $C \cup F \neq N$ **do**
     Pick $i \in N\backslash(C\cup F)$ with maximal $|S^{i,\mathrm{T}}| + |S^{i,\mathrm{T}}\cap F|$
     **If** $|S^{i,\mathrm{T}}| + |S^{i,\mathrm{T}}\cap F| = 0$
       **then** set $F = N\backslash C$
       **else** set $C = C\cup\{i\}$ and set $F = F\cup(S^{i,\mathrm{T}}\backslash C)$
     **endif**
```

The measure $|S^{i,\mathrm{T}}| + |S^{i,\mathrm{T}}\cap F|$ is purely heuristical. The first term is associated with the total number of strongly coupled neighbours, the second one with the number of strongly coupled neighbours which are in F. Domains with the same discretization stencil for most nodes (typically inner nodes) tend to have the same value of the measure $|S^{i,\mathrm{T}}| + |S^{i,\mathrm{T}}\cap F|$ for them. Note that the action to pick an index in step 2 of the above algorithm is non-deterministic and allows different implementations, depending on the chosen underlying data structures; see also Reference 41. Furthermore, using dynamic data structures and incremental techniques, it is possible to implement the overall set-up algorithm (i.e. phases I and II) to need a number of operations proportional to the number of fine grid unknowns. Further improvements should be possible if one were to handle nodes situated next to the boundary of the domain and inner nodes differently.

In a second phase the final C-point choice is made.

3.3. Selection of coarse grid points: set-up phase II

```
1. Set $T = \emptyset$
2. **While** $T \subset F$ **do**
     Pick $i \in F\backslash T$ and set $T = T \cup \{i\}$
     set $\tilde{C} = \emptyset$ and set $C^i = S_i \cap C$
     set $F^i = S_i \cap F$
     **While** $F^i \neq \emptyset$ **do**
        Pick $j \in F^i$ and set $F^i = F^i\backslash\{j\}$
        **If** $d(j, C^i)/d(i, \{j\}) \leqslant \beta$
           **then if** $|\tilde{C}| = 0$
              **then** set $\tilde{C} = \{j\}$ and set $C^i = C^i \cup \{j\}$
              **else** set $C = C \cup \{i\}$, set $F = F\backslash\{i\}$ and **goto 2**
           **endif**
        **endif**
     set $C = C \cup \tilde{C}$, set $F = F\backslash\tilde{C}$.
```

This second algorithm has to make sure that each point in F is strongly coupled directly with points in C or at least with points in F, which are strongly coupled with points in C. Again, the strategy to force the set $\tilde{C}$ to contain at most one element is purely heuristic. The parameters α and β which control the coarsening algorithm must be given by the user.

After the points N_L were divided into the sets F_L and C_L, we could define the interpolation as given in (25). In the algorithm of Ruge and Stüben[11,12] a little more sophisticated interpolation is used, which gives better results in numerical experiments:

$$(\mathscr{P}^L_{L-1}e_{L-1})_i := \begin{cases} (e_{L-1})_i, & i \in C_L, \\ -\dfrac{\sum\limits_{j\in C^i_L}((\mathscr{A}_L)_{ij} + c_{ij})(e_{L-1})_j}{(\mathscr{A}_L)_{ii} + c_{ii}}, & i \in F_L, \end{cases} \tag{27}$$

where

$$c_{ij} := \sum_{\substack{k\notin C^i_L \\ k\neq i}} \frac{(\mathscr{A}_L)_{ik}(\mathscr{A}_L)_{kj}}{(\mathscr{A}_L)_{ki} + \sum\limits_{l\in C^i_L}(\mathscr{A}_L)_{kl}}.$$

Once the interpolation matrix $\mathscr{P}^L_{L-1}$ is constructed, the system matrix $\mathscr{A}_{L-1}$ is determined by the Galerkin identity (23). Then the coarsening proceeds recursively until the number of remaining unknowns equals one.

3.4. Application to Navier–Stokes equations

In our algorithm we apply AMG for the solution of the potential equation for the initial velocity field (3) and of the Poisson equation for the pressure (8) in the explicit code or the pressure correction p' (16) in the SMAC code respectively. For these equations, a single set-up step is sufficient which consists of the set-up phases I and II in the initializing phase of the algorithm, because the equations differ only in the boundary values and the right-hand side, whereas the set-up depends only on the matrix of the linear system. For the SIMPLE scheme we would need a set-up step for the equation for the pressure correction (20) in each time step, because this equation depends on the velocities of the previous time step. The same holds for the SIMPLEC or SIMPLER scheme.

Moreover, we apply AMG to the momentum equation for $\vec{u}^*$ (11) and to the transport equation (22). Here we have to solve convection–diffusion problems where the convection dominates for high Reynolds numbers or low diffusion coefficients. The momentum equations change from time step to time step, because the time-dependent velocities $\vec{u}^{(n)}$ enter the scheme in the convective term. Thus we would need a set-up phase in each time step.

In our numerical experiments we also tried a sort of *adaptive set-up strategy* for the momentum equations. This means that we only apply the quite expensive set-up step if the number of AMG V-cycles exceeds a certain given number tol_{it} to reduce the residual below ε. Otherwise, we just keep the coarse grid and the interpolation operator of the previous time step. Thus the problems on the coarser grids are not exactly the problems related to the fine grid equation, but if the velocities do not change too much from one time step to the next, the coarse grid problem based on previous velocities might still produce sufficient coarse grid correction terms for the fine grid equations with the new velocities.

4. NUMERICAL RESULTS

In our numerical experiments we concentrate on the robustness of the algebraic multigrid method applied to the Navier–Stokes equations with respect to the geometry, the number of unknowns and the diffusion coefficient in the convection–diffusion equation. The accuracy of the discretization scheme is shown in Reference 18 for several examples such as the flow over a backward-facing step and the flow past an obstacle. Moreover, the AMG solver might be applied to some enhanced discretization schemes too.

In all our experiments we used a multigrid V-cycle with one pre- and post-smoothing step. As smoother we use Gauss–Seidel relaxation. We reduced the L^2-norm of the residuals to values below 10^{-12} and always measured the reduction rate, i.e. the quotient of the L^2-norms of the residuals of two successive iterates,

$$\rho_{it} := \frac{\|r_{it}\|_2}{\|r_{it-1}\|_2},$$

in the last iteration when the stopping criterion was reached. The AMG parameter β is always set to 0·35, whereas α is varied as given in the tables.

4.1. Dependence of convergence on geometry

To test the behaviour of our algorithm for complicated geometries, we consider a channel flow where several cubic obstacles are inserted in the channel, namely one, 2×2, 4×4, 8×8 and 16×16 cubes (see Figure 3). We used a 256×64 grid. The size of the cubes is chosen such that the sum of their volumes is constant for each test case.

First we study the potential equation (3) for the initial velocity field. Note that the equations for the pressure (8) in the explicit time-stepping scheme and the pressure correction (16) in the SMAC scheme are also Poisson equations with Neumann boundary conditions. They differ only in the right-hand side and the type of boundary conditions (inhomogeneous/homogeneous). Thus the convergence properties of AMG applied to those two equations are basically the same as for the potential problem (3).

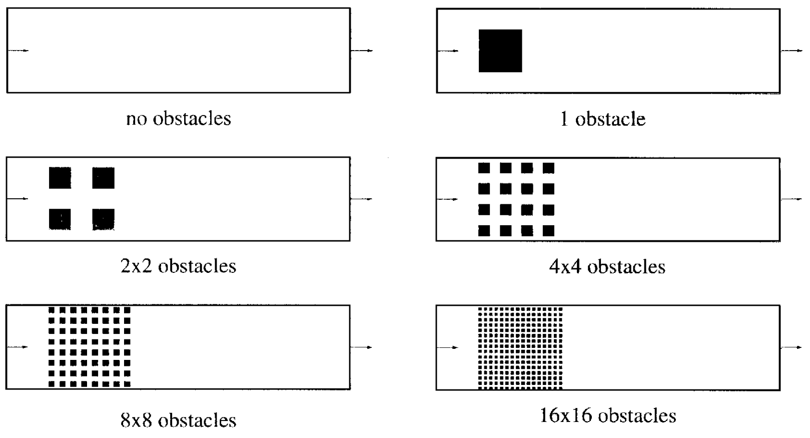

Figure 3. Test problem for dependence on geometry; flow from left to right

In Table I we see that the reduction rate depends strongly on the parameter α, which determines the set of strongly coupled neighbours of a point (see (26)). The best values (underlined) are obtained for $\alpha < 0{\cdot}1$. There the reduction rates were between 0·08 and 0·18. The minimal values for each geometry are always below 0·12, but they depend on α in a way which is not yet fully understood. Thus we can say that our algorithm is robust with respect to the geometry, but α must be chosen carefully. The number of coarse grid points is nearly the same for all cases, independent of the number of obstacles and the value of α. Thus the time which must be spent on one V-cycle is always in the same range.

We believe that the relatively bad reduction rates for larger values of α are caused by the Neumann boundary conditions or the semidefiniteness of the linear system. Also, fixing one point of the boundary data and thus obtaining a definite system does not improve the convergence rates. With pure Dirichlet conditions instead, we obtained much better reduction rates, also for large values of α, as can be seen in Table II.

The same results are obtained in the 3D case, where we considered a channel with $64 \times 32 \times 32$ grid cells and between zero and five obstacles in each direction (see Table III).

4.2. *Dependence of convergence on grid size*

Next we study the dependence of the reduction rates on the grid size. As test problem we took the example of Section 4.1 with one obstacle and varied the number of cells. We show the reduction rates for different values of α in Table IV (2D and 3D).

Table I. Dependence of reduction rate on number of obstacles and α; 2D potential equation with Neumann conditions

Obstacles	$\alpha = 1 \times 10^{-6}$	1×10^{-4}	0·01	0·02	0·03	0·05	0·1	0·15	0·25	0·45	0·75	0·95
0	0·108	0·112	0·094	0·100	$\underline{0·078}$	0·104	0·126	0·149	0·206	0·340	0·168	0·341
1	0·154	0·139	0·136	0·129	0·157	$\underline{0·108}$	0·164	0·166	0·289	0·216	0·295	0·662
2×2	$\underline{0·114}$	0·133	0·134	0·135	0·116	0·121	0·143	0·178	0·212	0·351	0·444	0·764
4×4	0·142	0·126	$\underline{0·110}$	0·129	0·111	0·124	0·176	0·169	0·193	0·352	0·509	0·789
8×8	0·148	0·118	$\underline{0·116}$	0·152	0·161	0·161	0·150	0·202	0·253	0·463	0·468	0·774
16×16	0·117	0·138	0·174	$\underline{0·112}$	0·122	0·147	0·173	0·136	0·251	0·372	0·488	0·838

Table II. Dependence of reduction rate on number of obstacles and α 2D potential equation with Dirichlet conditions

Obstacles	$\alpha = 1 \times 10^{-4}$	0·05	0·25	0·45	0·85
0	0·081	0·079	0·099	0·101	0·035
1×1	0·088	0·066	0·095	0·061	0·035
2×2	0·095	0·067	0·101	0·089	0·033
4×4	0·090	0·080	0·100	0·058	0·033
8×8	0·080	0·079	0·100	0·081	0·149
16×16	0·082	0·079	0·098	0·128	0·091

Table III. Dependence of reduction rate on number of obstacles and α 3D potential equation with Neumann conditions

Obstacles	$\alpha = 1 \times 10^{-6}$	1×10^{-4}	0·01	0·02	0·03	0·05	0·10	0·15	0·25	0·45	0·75	0·95
0	0·086	0·101	0·102	0·088	0·080	0·170	0·119	0·164	0·275	0·390	0·438	0·486
1	0·140	0·160	0·116	0·134	0·131	0·116	0·136	0·178	0·309	0·480	0·551	0·687
$2 \times 2 \times 2$	0·190	0·188	0·133	0·134	0·127	0·154	0·121	0·167	0·346	0·447	0·559	0·694
$3 \times 3 \times 3$	0·160	0·109	0·117	0·099	0·124	0·170	0·139	0·300	0·491	0·616	0·690	0·690
$4 \times 4 \times 4$	0·094	0·112	0·147	0·110	0·121	0·217	0·152	0·189	0·334	0·541	0·583	0·706
$5 \times 5 \times 5$	0·103	0·139	0·111	0·112	0·115	0·133	0·149	0·201	0·289	0·396	0·533	0·762

Table IV. Dependence of reduction rate on grid size and α; potential equation with Neumann conditions, 2D and 3D

Grid	$\alpha = 1 \times 10^{-6}$	1×10^{-4}	0·01	0·02	0·03	0·05	0·1	0·15	0·25	0·45	0·75	0·95
64×16	0·095	0·095	0·064	0·104	0·137	0·128	0·095	0·129	0·147	0·251	0·108	0·557
128×32	0·084	0·090	0·133	0·147	0·115	0·128	0·161	0·153	0·157	0·362	0·099	0·658
256×64	0·154	0·139	0·136	0·129	0·157	0·108	0·164	0·166	0·289	0·216	0·295	0·662
512×128	0·142	0·115	0·144	0·144	0·312	0·137	0·151	0·257	0·259	0·443	0·263	0·727
$32 \times 16 \times 16$	0·086	0·104	0·095	0·109	0·137	0·148	0·124	0·123	0·156	0·229	0·434	0·723
$64 \times 32 \times 32$	0·140	0·160	0·116	0·134	0·131	0·116	0·136	0·178	0·309	0·480	0·551	0·687
$96 \times 32 \times 32$	0·137	0·136	0·159	0·160	0·149	0·145	0·180	0·176	0·182	0·533	0·583	0·638

Table V. Dependence of reduction rate on grid size and α; potential equation with Dirichlet conditions, 2D and 3D

Grid	$\alpha = 1 \times 10^{-4}$	0·05	0·25	0·45	0·85
64×16	0·062	0·050	0·052	0·039	0·017
128×32	0·070	0·057	0·075	0·048	0·023
256×64	0·088	0·066	0·095	0·061	0·035
512×128	0·024	0·068	0·069	0·086	0·076
$32 \times 16 \times 16$	0·0384	0·0420	0·0383	0·0437	0·0623
$64 \times 32 \times 32$	0·0518	0·0510	0·1050	0·0870	0·1622
$96 \times 32 \times 32$	0·0571	0·0510	0·0822	0·2992	0·2991

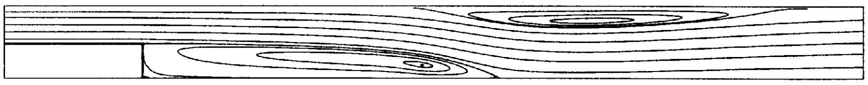

Figure 4. Flow over a backward-facing step; streamlines, $Re = 500$

We have a strong dependence on α, with the best results obtained for $\alpha < 0{\cdot}1$. Here the convergence rates increase slightly for larger grids.

However, we consider the potential equation with Dirichlet conditions (Table V), we obtain convergence rates which are in the same range, independent of the grid size. Thus the relatively bad convergence behaviour must be caused by the Neumann conditions.

4.3. Dependence of convergence on diffusion coefficient

Now we consider the dependence of the convergence properties of the AMG algorithm for convection–diffusion equations such as the momentum equations (11) as they appear in the SMAC and SIMPLE schemes and the transport equation (21). This is also a problem where standard multigrid methods fail. As test problem we take the flow over a backward-facing step with Reynolds number $Re = 500$ (see Figure 4). Here two recirculating regions appear.

We consider the convergence behaviour of our AMG algorithm for the transport equation (21) with Dirichlet boundary conditions on the left side and Neumann (adiabatic) boundary conditions on the remaining three sides. In 2D we employed a mesh with 300×75 cells and in 3D we used an $80 \times 16 \times 16$ grid. As already mentioned, the time-discrete transport equation (22) is of the same type as the momentum equations for each component of the tentative velocity (11). The results are shown in Tables VI and VII. For 2D we also present the complexity of the coarse grids (Comp.), i.e. the number of unknowns on all levels divided by the number of unknowns on the finest level, and the connectivity of the coarse grid operators (Conn.), i.e. the number of non-zero entries in the matrices on all levels divided by those on the finest level. Thus both numbers indicate the work time necessary for one multigrid iteration.

Table VI. Dependence of reduction rate (red.), complexity (Comp.) and connectivity (Conn.) on diffusion parameter λ and α; transport equation, backward-facing step, 2D

λ		$\alpha = 1 \times 10^{-4}$	0·01	0·05	0·15	0·25	0·45	0·85
1	Red.	0·081	0·062	$\underline{0{\cdot}043}$	0·091	0·079	0·089	0·152
	Comp.	1·874	2·035	1·831	1·827	1·887	1·900	1·944
	Conn.	4·368	5·349	3·360	2·752	2·968	3·024	3·083
10^{-2}	Red.	$\underline{0{\cdot}025}$	0·040	0·032	0·045	0·066	0·105	0·406
	Comp.	1·884	1·924	1·995	1·967	1·950	1·969	1·960
	Conn.	4·483	4/515	4·688	3·735	3·265	3·358	2·841
10^{-4}	Red.	$\underline{0{\cdot}004}$	0·007	0·050	0·092	0·107	0·114	0·164
	Comp.	2·264	2·312	2·179	2·078	2·032	2·031	1·989
	Conn.	10·01	6·952	5·051	3·571	3·127	2·688	2·311
10^{-6}	Red.	$\underline{0{\cdot}003}$	0·004	0·012	0·058	0·085	0·101	0·192
	Comp.	2·327	2·270	2·082	2·057	2·039	2·021	1·978
	Conn.	9·337	5·585	4·000	3·120	2·899	2·566	2·266
10^{-8}	Red.	$\underline{0{\cdot}002}$	0·004	0·012	0·066	0·084	0·091	0·184
	Comp.	2·317	2·250	2·076	2·047	2·029	2·009	1·967
	Conn.	9·417	5·591	3·863	3·183	2·882	2·539	2·249

Table VII. Dependence of reduction rate on diffusion parameter λ and α; transport equation, backward-facing step, 3D

λ	$\alpha = 1 \times 10^{-4}$	0·01	0·05	0·25	0·45	0·85
1	0·106	$\underline{0{\cdot}097}$	0·125	0·135	0·470	0·624
10^{-2}	0·027	$\underline{0{\cdot}027}$	0·030	0·221	0·235	0·364
10^{-4}	$\underline{0{\cdot}002}$	0·008	0·034	0·108	0·124	0·142
10^{-6}	$\underline{5 \times 10^{-4}}$	0·005	0·036	0·105	0·115	0·135
10^{-8}	$\underline{6 \times 10^{-4}}$	0·005	0·037	0·103	0·111	0·135

We see that also for the convection–diffusion equation we get the best reduction rates ($\rho < 0{\cdot}1$, for some α even below 0·01) for $\alpha \leqslant 0{\cdot}15$, independent of λ. However, for very small α and especially small λ the connectivity is worse than for larger values of α. Note that the recirculating regions do not effect the convergence numbers.

Moreover, we consider the dependence of the convergence rate for the transport equation on the number of obstacles and the diffusion coefficient. Therefore we again choose the domain with cubic obstacles as shown in Figure 3 and set the AMG parameter $\alpha = 0{\cdot}05$. As we can see in Table VIII, the convergence rates do not vary very much for a fixed diffusion coefficient and different numbers of obstacles. Interestingly, the rates become better for smaller diffusion coefficients, but even for $\lambda = 1$ they are still quite good.

Equivalent results are obtained if we consider the dependence of the reduction rates for the transport equation on the grid size and the diffusion coefficient (see Table IX).

4.4. Full Navier–Stokes solver

Now we compare the solution process of the Navier–Stokes equations by the explicit and the SMAC semi-implicit scheme using AMG for the computation of the initial velocity field, for the

Table VIII. Dependence of reduction rate on number of obstacles and λ; transport equation, 2D, $\alpha = 0{\cdot}05$, 256×64 cells

Obstacles	$\lambda = 1$	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
0	0·053	0·047	0·077	0·0008	0·0007	0·0007
1	0·077	0·046	0·065	0·025	0·027	0·027
2×2	0·086	0·030	0·063	0·051	0·053	0·057
4×4	0·097	0·038	0·054	0·036	0·027	0·023
8×8	0·133	0·049	0·071	0·033	0·027	0·027
16×16	0·147	0·048	0·060	0·057	0·031	0·031

Table IX. Dependence of reduction rate on grid size and λ; transport equation, 2D, $\alpha = 0{\cdot}05$, one obstacle

Grid	$\lambda = 1$	10^{-2}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
64×64	0·047	0·013	0·027	0·013	0·013	0·014
128×32	0·069	0·038	0·022	0·025	0·025	0·025
256×64	0·077	0·046	0·065	0·025	0·027	0·027
512×128	0·077	0·067	0·079	0·079	0·033	0·033

pressure in the explicit scheme and the pressure correction as the tentative velocities in the semi-implicit scheme. Here we also applied the above-mentioned adaptive set-up strategy.

As test problem we consider the flow around an obstacle at $Re = 20$ and use a mesh with 220×41 cells in 2D and $60 \times 12 \times 12$ cells in 3D. We run our programme until $t = 15$ was reached. We stopped the iterations if the norm of the residual was below 10^{-6}. The explicit time-stepping scheme becomes unstable for time step sizes $\delta t > 0{\cdot}014$ in 2D and $\delta t > 0{\cdot}051$ in 3D, whereas the semi-implicit code still shows good results for $\delta t = 0{\cdot}5$ in both 2D and 3D. However, we see in Tables X and XI that the time spent on the computation of one time step in the semi-implicit code is much larger than for the explicit code. This is due to the time which must be spent on the set-up phase. Nevertheless, because of the large number of allocation and comparison steps, the time spent on this phase depends a lot on the hardware platform available.*

The time spent for the semi-implicit code can be reduced if we use the adaptive set-up strategy. Here we performed a new set-up step only if the number of iterations in the previous time step was greater than or equal to the given value tol_{it}. Otherwise, the coarse grid correction computed with the coarse grid sequence of the previous time step is still good enough to reduce the error in the new time step efficiently. As we see in Tables X and XI, there are dramatic differences in the computing times for $\delta t = 0{\cdot}05$ and $tol_{it} = 1$ (set-up in every time step) and $tol_{it} = 4$ (where only one set-up step is needed for each momentum equation), whereas the number of iterations is not much larger.

Thus the semi-implicit algorithm using adaptive set-up is quite efficient compared with the explicit algorithm, because the size of the time step is not so restricted. We assume that the difference is more severe for finer grids, but here we must also mention that the memory needed for the semi-implicit algorithm is much larger, because the matrices for the momentum equations including the coarse grid operators must be stored.

Table X. Number of set-up steps, number of iterations and overall computation time for Navier–Stokes solver; $Re = 20$, 2D

Scheme	tol_{it}	δt	Time steps	u-Set-up	v-Set-up	u-Iter.	v-Iter.	Comp. time
Explicit	—	0·01	1500	—	—	—	—	14 min 05·25 s
Semi-implicit	4	0·01	1500	1	1	2226	1976	99 min 06·72 s
Semi-implicit	4	0·02	750	1	1	1230	1098	50 min 28·67 s
Semi-implicit	1	0·05	300	300	300	514	459	10 min 53·13 s
	4	0·05	300	1	1	560	483	23 min 22·48 s
Semi-implicit	4	0·10	150	3	3	293	257	12 min 40·28 s
Semi-implicit	3	0·20	75	23	43	174	194	17 min 05·46 s
	4	0·20	75	4	4	188	199	8 min 36·73 s
	5	0·20	75	2	2	213	209	8 min 11·94 s
Semi-implicit	3	0·50	30	30	30	94	92	11 min 39·24 s
	4	0·50	30	9	6	99	95	5 min 51·10 s
	5	0·50	30	5	4	109	104	5 min 06·77 s
	10	0·50	30	2	2	149	130	4 min 51·65 s

* We used an ~HP 9000/712 workstation.

Table XI. Number of set-up steps, number of iterations and overall computation time for Navier–Stokes solver; $Re = 20$, 3D

Scheme	tol_{it}	δt	Time steps	u-Set-up	v-Set-up	w-Set-up	u-Iter·	v-Iter.	w-Iter.	Comp. time
Explicit	—	0·03	500	—	—	—	—	—	—	1 h 00 min 57·40 s
Semi-implicit	4	0·03	500	1	1	1	900	836	724	2 h 17 min 6·40 s
Semi-implicit	4	0·06	250	1	1	1	497	479	423	1 h 16 min 22·69 s
Semi-implicit	1	0·10	150	150	150	150	308	286	242	10 h 16 min 43·39 s
	4	0·10	150	3	2	1	316	305	279	0 h 55 min 16·78 s
Semi-implicit	3	0·20	75	35	30	15	182	169	142	2 h 05 min 17·56 s
	4	0·20	75	7	4	2	187	177	157	0 h 40 min 57·27 s
	5	0·20	75	2	2	1	202	185	166	0 h 32 min 20·19 s
Semi-implicit	3	0·50	30	30	30	29	110	119	92	1 h 55 min 07·15 s
	4	0·50	30	18	27	4	110	119	94	1 h 04 min 36·13 s
	5	0·50	30	4	3	2	127	124	114	0 h 23 min 01·20 s
	10	0·50	30	2	2	2	140	137	119	0 h 19 min 45·11 s

4.5. Two other problems with complicated domains

Finally, we present some results of two other problems with complicated geometries, namely the flow through a river system, here the delta of the Ganges in Bangladesh,* and the flow through a porous medium on a microscale level.

In Figures 5 and 7 we show the geometric structure of the computational domains and the stationary velocity fields. For the Ganges example we use inflow conditions at the five branches at the top. For the porous medium example we use inflow on the left and outflow on the right.

Figures 6 and 8 show the propagation in time of a chemical pollution modelled by the transport equation (21). In the Ganges example the permanent source of pollution is situated at the top of the second branch from the left. In the porous medium example the pollution enters at the middle of the left boundary.

The convergence properties are in the same range as for the test problems reported above.

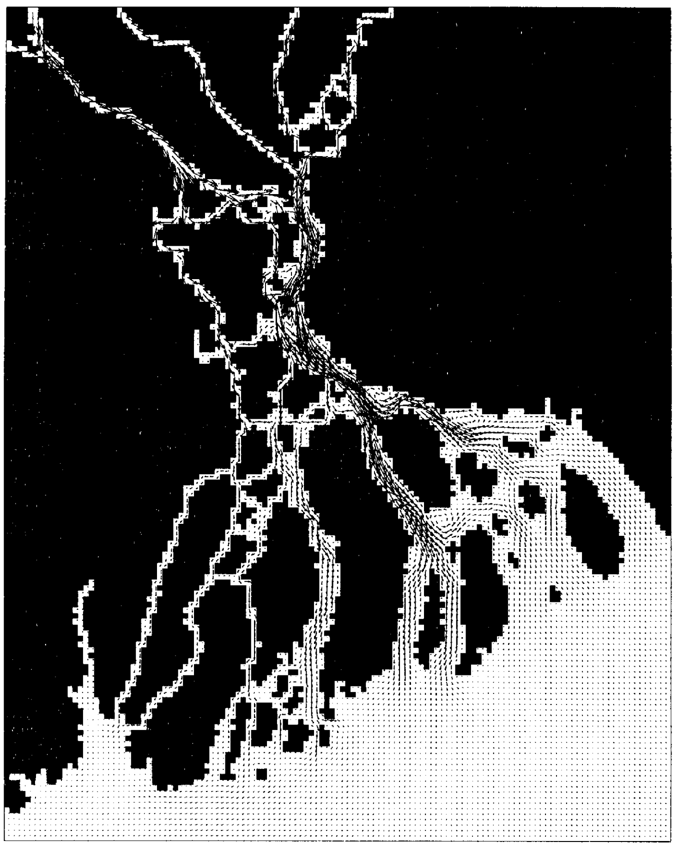

Figure 5. Ganges delta; velocity plot

* Note that this two-dimensional calculation is not realistic at all. First, the resolution of the domain is far too coarse; second, the third dimension, i.e. the deepness of the river arms, is not involved, which influences the computed flow velocity; third, realistic inflow and outflow conditions are not known. This is only an example for complicated geometries.

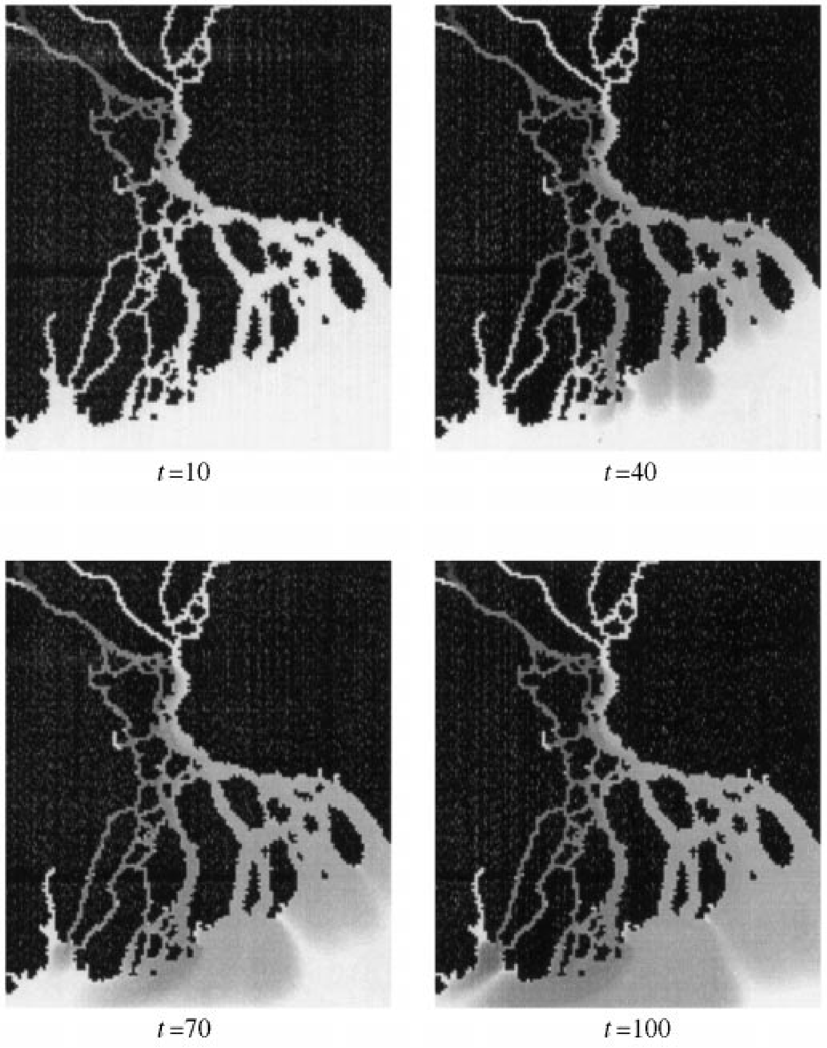

Figure 6. Pollution transport in Ganges delta; $\lambda = 4{\cdot}6 \times 10^{-11}$

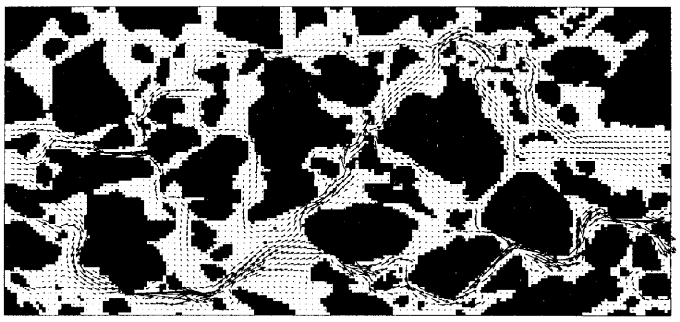

Figure 7. Porous medium; velocity plot

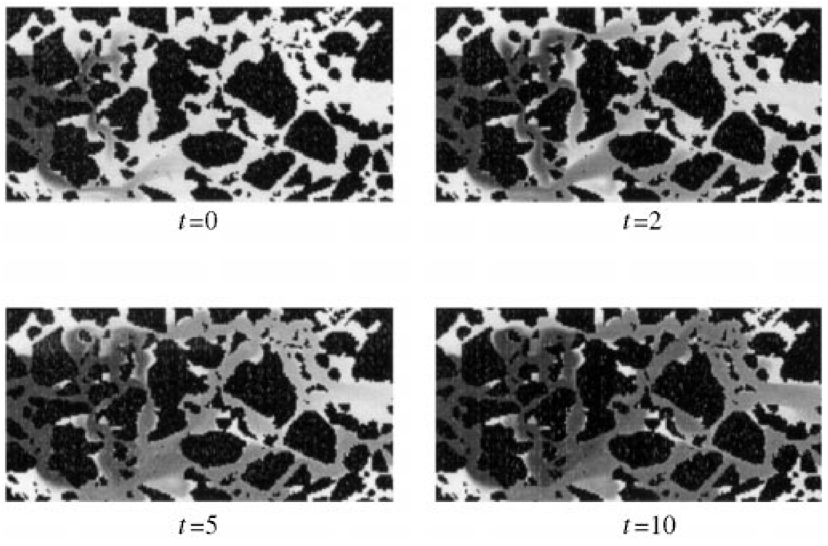

Figure 8. Pollution transport in porous medium; $\lambda = 10^{-4}$

5. CONCLUSIONS

In this paper we considered the application of algebraic multigrid methods to the Poisson equation and the convection–diffusion equation in complicated geometries. Both equations arise in the numerical solution of the Navier–Stokes equations using explicit or semi-implicit time discretization.

For equations in complicated geometries and for convection-dominated problems the convergence rates of standard multigrid methods usually deteriorate. In several numerical experiments we demonstrated that the application of AMG leads to robust and efficient algorithms, especially for a proper choice of the AMG parameter α which controls the coarsening process. However, the dependence of the convergence rate on α is not yet fully understood. Moreover, a modification of the algorithm for Neumann boundary conditions might improve the convergence rates for the pressure equation.

Furthermore, our experiments show that an explicit time-stepping scheme, where we have a strong restriction on the time step size, can still compete with the semi-implicit algorithms in terms of run time. This is due to the large amount of time spent on the AMG set-up phases.

However, we conclude that the semi-implicit time discretization is superior for finer grids where the time step size in the explicit scheme must be reduced more and more to preserve stability. This holds especially for stationary problems, where we can apply an adaptive set-up strategy in which the set-up is not done in every time step but only if the convergence rate becomes worse than a prescribed value.

ACKNOWLEDGEMENTS

We are grateful to Erwin Wagner, Ralph Kreissl and Markus Rykaschewski who implemented and tested the 3D version of the AMG Navier–Stokes solver.

REFERENCES

1. F. J. Fayers and T. A. Hewett, ‘A review of current trends in petroleum reservoir description and assessing the impacts on oil recovery’, in T. F. Russell (ed.), *Computational Methods in Water Resources IX*, Vol. 2, *Mathematical Modeling in Water Resources*, Computational Mechanics Publications, Southampton, 1992, pp. 3–33.
2. R. Kettler, ‘Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods’, *Lect. Notes Math.*, **960**, 502–534 (1982).
3. G. Wittum, ‘Linear iterations as smoothers in multigrid methods: theory with applications to incomplete decompositions’, *Impact Comput. Sci. Engng.*, **1**, 180–215 (1989).
4. G. Wittim, ‘On the robustness of ILU0-smoothing’, *SIAM J. Sci. Stat. Comput.*, **10**, 699–717 (1989).
5. P. M. de Zeeuw, ‘Matrix-dependent prolongations and restrictions in a blackbox multigrid solver’, *J. Comput. Appl. Math.*, **33**, 1–27 (1990).
6. A. Reusken, ‘Multigrid with matrix-dependent transfer operators for convection–diffusion problems’, in P. W. Hemker and P. Wesseling (eds), *Multigrid Methods IV*, Birkhäuser, Basle, 1994, pp. 269–280.
7. R. Bank and J. Xu, ‘A hierarchical basis multi-grid method for unstructured grids’, in W. Hackbusch and G. Wittum (eds), *Fast Solvers for Flow Problems, Proc. 10th GAMM Seminar*, Vieweg, Braunschweig, 1995.
8. R. Bank and J. Xu, ‘An algorithm for coarsening unstructured meshes’, *Numer. Math.*, **73**, 1–36 (1996).
9. W. Hackbusch and S. Sauter, ‘Composite finite elements for the approximation of pdes on domains with complicated micro-structures’, *Numer. Math.*, **75**, 447–472 (1995).
10. R. Kornhuber and H. Yserentant, ‘Multilvel-methods for elliptic problems on domains not resolved by the coarse grid’, *Contemp. Math.*, **180**, 49–60 (1994).
11. J. Ruge and K. Stüben, ‘Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG)’, *Arbeitspapiere der GMD 89*, 1984.
12. J. Ruge and K. Stüben, ‘Algebraic multigrid (AMG)’, *Arbeitspapiere der GMD 210*, 1986.
13. T. Grauschopf, M. Griebel and H. Regler, ‘Additive multilevel-preconditioners based on bilinear interpolation matrix-dependent geometric coarsening and algebraic multigrid coarsening for second order elliptic pdes’, *Appl. Numer. Math.*, **23**, (1997).
14. R. D. Lonsdale, ‘An algebraic multigrid solver for the Navier–Stokes equations on unstructured meshes’, *Int. J. Numer. Meth. Heat Fluid Flow*, **3**, 3–14 (1993).
15. R. Webster, ‘An algebraic multrigrid solver for Navier–Stokes problems, *Int. j. numer. meth. fluids*, **18**, 761–780 (1994).
16. M. Raw, ‘A coupled algebraic multigrid method for the 3D Navier–Stokes equations’, *Tech. Rep.*, Advanced Scientific Computing, Waterloo, Ont., 1994.
17. C. W. Hirt, B. D. Nichols and N. C. Romero, ‘Sola—a numerical solution algorithm for transient fluid flows’, *Los Alamos Scientific Lab. Rep. LA-5852*, 1975.
18. M. Griebel, T. Dornseifer and T. Neunhoeffer, *Numerical Simulation in Fluid Dynamics—A Practical Introduction*, SIAM Philadelphia, 1998.
19. L. Quartapelle, *Numerical Solution of the Incompressible Navier–Stokes Equations*, ISNM Vol. 113, Birkhäuser, Basle, 1993.
20. F. H. Harlow and J. E. Welch, ‘Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface’, *Phys. Fluids*. **8**, 2182–2189 (1965).
21. T. Hughes, L. Franca and M. Balestra, ‘A new finite element formulation for computational fluid dynamics: III. The general streamline operator for multidimensional advective–diffusion systems’, *Comput. Meth. Appl. Mech. Engng.*, **58**, 305–328 (1986).
22. C. Johnson, *Numerical Solutions of Partial Differential Equations by the Finite Element Method*, Cambridge University Press, Cambridge, 1987.
23. S. Turek, ‘A comparative study of some time-stepping techniques for the incompressible Navier–Stokes equations’, *Preprint 95-10*, IWR Heidelberg, 1995.
24. C. Nonino and S. del Giudice, ‘An improved procedure for finite-element methods in laminar and turbulent flow ’ in C. Taylor (ed.), *Numerical Methods in Laminar and Turbulent Flow, Part 1*, Pineridge, Swansea, 1985, pp. 597–608.
25. Y. M. Kim and T. J. Chung, ‘Finite-element analysis of turbulent diffusion flames’, *AIAA J.*, **27**, 330–339 (1988).
26. L. Cheng and S. Armfield, ‘A simplified marker and cell method for unsteady flows on non-staggered grids’, *Int. j. numer. meth. fluids*, **21**, 15–34 (1995).
27. A. Amsden and F. H. Harlow, ‘‘The SMAC method: a numerical technique for calculating incompressible fluid flow’, *Los Alamos Scientific Lab. Rep. LA-4370*, 1970.
28. S. V. Patankar and D. B. Spalding, ‘A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows’, *Int. J. Heat Mass Transfer*, **15**, 1787–1806 (1972).
29. S. V. Patankar, ‘A calculation procedure for two-dimensional elliptic situations’, *Numer. Heat Transfer*, **4**, 409–425 (1981).
30. S. V. Patankar, *Numerical Heat Transfer and Fluid Flow*, McGraw-Hill, New York, 1980.
31. A. J. Chorin, ‘Numerical solution of the Navier–Stokes equations’, *Math. Comput.*, **22**, 745–762 (1968).
32. R. Temam, ‘Sur l’approximation de la solution des équations de Navier–Stokes par la méthode des pas fractionnaires’, *Arch. Rat. Mech. Anal.*, **32**, 135–153 (1969).
33. R. I. Issa, ‘Solution of the implicitly discretized fluid flow equations by operator-splitting’, *J. Comput. Phys.*, **62**, 40–65 (1986).

34. J. van Kan, 'A second-order accurate pressure correction scheme for viscous incompressible flow', *SIAM J. Sci. Stat. Comput.*, **7**, 870–891 (1986).
35. S. W. Kim and T. J. Benson, 'Comparison of the SMAC, PISO and iterative time-advancing schemes for unsteady flow', *Comput. Fluids*, **21**, 435–454 (1992).
36. A. Oberbeck, 'Über die Wärmeleitung der Flüssigkeiten bei Berücksichtigung der Strömungen infolge von Temperaturdifferenzen', *Ann. Phys. Chem.*, **7**, 271–292 (1879).
37. J. Boussinesq, *Theorie Analytique de la Chaleur*, Vol. 2, Gauthier-Villars, Paris, 1903.
38. A. Bejan, *Convection Heat Transfer*, Wiley–Interscience, New York, 1984.
39. A. Brandt, S. McCormick and J. Ruge, 'Algebraic multigrid (AMG) for automatic algorithm design and problem solution', Refs. Colorado State University, Ft. Collins, CO, 1982.
40. A. Brandt, 'Algebraic multigrid theory: the symmetric case' in S. McCormick and U. Trottenberg (eds), *Prelim. Proc. Int. Multigrid Conf.*, Copper Mountain, CO, April 1983.
41. H. Bungartz, 'Beschränkte Optimierung mit algebraischen Mehrgittermethoden', *Diplomarbeit*, Institut für Informatik, Technische Universität München, 1988.