Computing 55, 379-393 (1995)

Towards Algebraic Multigrid for Elliptic Problems of Second Order

Dietrich Braess, Bochum

Abstract — Zusammenfassung

Towards Algebraic Multigrid for Elliptic Problems of Second Order. An algebraic multigrid method is developed which can be used as a preconditioner for the solution of linear systems of equations with positive definite matrices. The method is directed to equations which arise from the discretization of elliptic equations of second order, but only the matrix is the source for the information used by the algorithm. One has only to know whether the matrix stems from a 2-dimensional or 3-dimensional problem and whether the elliptic equations are scalar equations or belong to a system.

AMS Subject Classification: 65N20.

Key words: Algebraic multigrid, preconditioning.

Ein algebraisches Mehrgitterverfahren für elliptische Probleme zweiter Ordnung. Es wird ein algebraisches Mehrgitterverfahren vorgestellt, das zur Vorkonditionierung von positiv definiten Matrizen geeignet ist. Es wurde entworfen für Gleichungssysteme, die aus der Diskretisierung von elliptischen Differentialgleichungen stammen. Alle Information wird aus der Matrix herausgezogen. Man braucht nur zu wissen, ob ein zwei- oder dreidimensionales Problem und ob eine skalare Gleichung oder ein System zugrunde liegt.

1. Introduction

Finite element problems often lead to large systems of linear equations with positive definite symmetric matrices. In many cases there are efficient solvers available. If for instance the system is only of medium size and if the coefficients of the underlying partial differential equations do not vary too much, then the method of conjugate gradients with standard preconditioners is a good tool. If, on the other hand, the mesh is regular or if it is generated by successive refinements, then multigrid methods are very efficient.

There are problems, however, for which the two methods have serious shortcomings. These are problems with unstructured grids. Typical are those which we encounter in groundwater simulation. In realistic models, the grid points and meshes are chosen according to topographical and geological data. The grids are not refinements of some coarser ones. Therefore, it is not easy to treat the given meshes with existing multigrid algorithms. Moreover, an ad hoc construction of coarse grids would imply here that the coefficients of the pde's vary by factors of more than 100 within some elements of the coarse grids. On the other hand, the big variations of the coefficients and the large number of unknowns imply that the condition numbers are high and that cg-methods with standard preconditioners are close to breakdown.

In these cases there is a demand for efficient algorithms which require only the matrix of the linear system, but no detailed information on the underlying unstructured grid. Approaches into this direction can be found in the "Algebraic multigrid method" by Ruge and Stüben [9, 10, 11] and the "Black box multigrid" by de Zeeuw [13]. These concepts were however restricted to M-matrices or to regular grids. On the other hand R. E. Bank [1] refers to variational problems, but he requires the knowledge of the coordinates of the grid points which correspond to the variables. Also the concept of aggregation by Chatelin and Miranker [4] is to be mentioned in this framework.

We have started another approach towards algebraic multigrid. We assume only that the matrices are symmetric and positive definite, they need not be M-matrices. Our algorithm extracts almost all the given information from the matrix. One merely has to know whether the underlying domain is of dimension two or three and whether we have a scalar equation or a system.

The main idea is the construction of simple coarse grid functions which can live on unstructured grids. Roughly speaking, the variables are put into groups of 3 to 4 nodes. The subspace which may be understood as the subspace of coarse grid functions, contains those elements which attain the same value at the nodes which belong to the same group, c.f. [2].

The resulting multigrid procedure is used as a preconditioner. It is not so efficient as the well established multigrid codes for structured grids, but it has a good performance with matrices which are found in applications of actual interest. The main advantage is the use as a black-box method so that only the matrices need to be known, and the grid is arbitrary.

A FORTRAN code of the algorithm may be requested from the author.

2. Construction of Coarse Grids

Our aim is an algorithm which makes use only of the given matrix. Nevertheless, we will refer to matrix-vector notation very seldom in our construction. We rather identify the vectors in n-space in the usual way with real-valued functions on n points. Two points p_i and p_j are neighbours or adjacent if we have $a_{ij} \neq 0$ for the associated matrix element. So an abstract graph is defined. In general it will not be planar. We note that this graph differs from the grid of the finite element triangulation, although there is some resemblance. It is a ficticious grid in the terminology of [11].

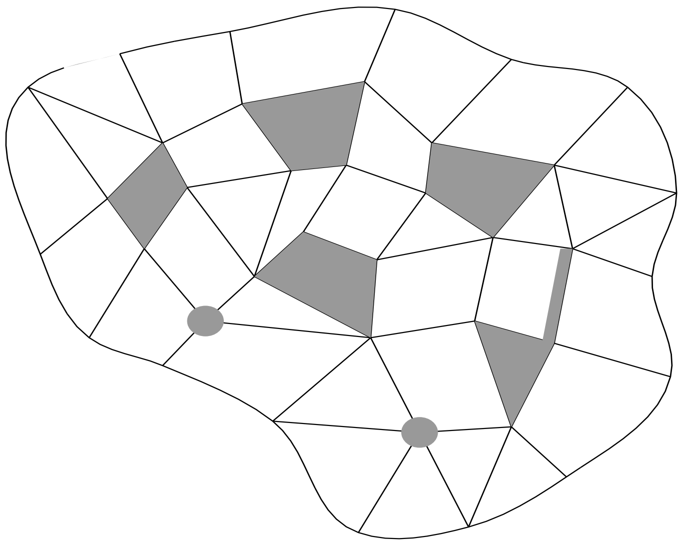

Fig. 1. Mesh with 22 interior nodes made of triangles and quadrilaterals. The shaded regions refer to the coarse grid.

First we consider matrices which stem from variational problems for scalar functions on two-dimensonal domains.

Assume that the given linear system of equations arises from a finite element calculation with the grid depicted in Fig. 1. The unstructured grid contains triangles and quadrilaterals, and the finite element functions are linear and bilinear, respectively. There is little hope to construct a subspace of functions which are linear or bilinear on a subnet.

Therefore, we will construct in a different way a subspace whose elements will play the role of coarse grid functions. *We partition the nodes into groups with up to four nodes in each group. Those functions which attain the same value at the points of each group, will be considered as coarse grid functions.* The restriction to this subspace leads to a variational problem of lower dimension.

Once we have chosen the groups, the stiffness matrix for the reduced variational problem, say A^c, is easily computed from the original stiffness matrix A. Let $I_1, I_2, \dots, I_m$ define the partitioning, specifically let I_j contain the numbers of the points which belong to the j-th group [2]. We have

$$A^c_{j\ell} = \sum_{i \in I_j} \sum_{k \in I_\ell} A_{ik} \quad \text{for } j, \ell = 1, 2, \dots, m.$$

Obviously the dimension of the reduced problem equals m.

The matrix representations of the prolongation and restriction operators contain only zeros and ones. This was done since this choice leads to coarse grid matrices with a small number of nonzero entries, c.f. the remark on unrealistic AMG solvers in [9]. On the other hand we cannot expect good results in those cases in which other authors suggest matrix dependent prolongations or restrictions [12, 13].

A suitable partitioning is obvious for a regular rectangular grid. Its construction is more involved for unstructured grids. Fig. 1 shows a partition of the 22 interior nodes into 7 groups. Four of them refer to quadrilaterals, one to another configuration with 4 nodes, and two contain only one node.

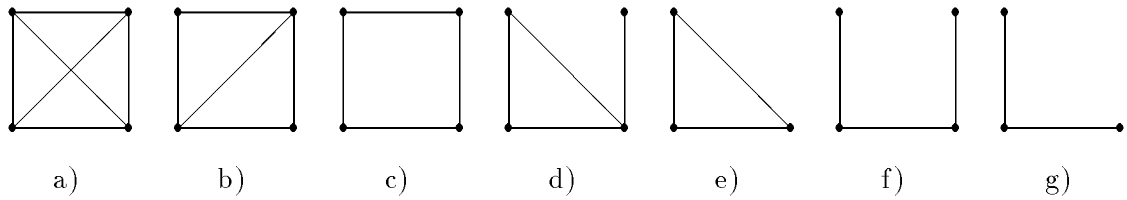

Fig. 2. Patterns in the construction of groups ordered due to the preference.

We have tested two methods. The first one is based only on the position of the nonzero entries of the matrix and not on their size.

When constructing the groups, we try to find patterns in the graph of the form shown in Fig. 2. The patterns are depicted in the order of their priority, i.e., the pattern in 2a has the highest and that one in 2g the lowest priority.

The groups are created successively. New groups are constructed in two steps. First, two elements of the new group are chosen such that they and two elements of an existing group together make a pattern of the form which we know from Fig. 2. In a second step they are completed to a pattern with the patterns of high priority preferred.

Algorithm for the partitioning of the variables into groups.
Choose one group G_1 of 4 variables which form a feasible pattern;
$i = 1$;
repeat
  let $G_1, G_2, \ldots, G_p$ be the already constructed groups;
  let R be the complement of $G_1 \cup G_2 \cup \ldots \cup G_p$;
  set $i = \min\{j \geq i;$ an element in G_j has a neighbour in $R\}$;
  if there is a feasible pattern with 2 elements in G_i and 2 elements in R
  then
    determine the best pattern of this kind;
    let j, k be the 2 elements of this pattern which belong to R;
    determine the best pattern in R which contains j, k
    [Here, the trivial pattern with only j and k is admitted,
    and the constructed pattern may contain 2, 3 or 4 elements];
    put the elements of the constructed pattern into G_{p+1}
  else

choose up to 3 points from R which are neighbours of G_i;
put these elements into G_{p+1};
end if
until no element of $G_1, G_2, ..., G_{p+1}$ has a neighbour in the complement
end. $\square$

The second method for the construction of groups proceeds in two steps. In the first step pairs are formed (as far as possible). In this process a node is considered free as long as it does not belong to a pair. For each free node i a free node j is determined such that

$$\frac{a_{ij}^2}{a_{ii}a_{jj}}$$

is maximized. The quotient is a measure of the coupling. In the second step for each pair another pair is searched such that a pattern as best as possible in the sense of Fig. 2 is created. So geometrical properties and the strength of the coupling enter into the construction. Arguments for the combination will be given in the next section.

Algorithm for the *alternative* partitioning of the variables into groups.
(*Building of pairs*)
repeat
let $P_1, P_2, ..., P_p$ be the already constructed pairs;
let R be the complement of $P_1 \cup P_2 \cup ... \cup P_p$;
choose a node $i \in R$;
if there is a neighbour of i in R
determine the neighbour $j \in R$, for which $\dfrac{a_{ij}^2}{a_{ii}a_{jj}}$ is maximal;
set $P_{p+1} = \{i, j\}$
else
set $P_{p+1} = \{i\}$
end if
until the complement of $P_1 \cup P_2 \cup \ldots \cup P_{p+1}$ is empty;
(*Building of groups*)
repeat
let $G_1, G_2, ..., G_p$ be the already constructed groups;
let R be the set of those pairs which have not combined to groups;
choose a pair $P_i \in R$;
define $m_{ij} =$ number of $\{A_{k\ell} \neq 0; k \in P_i, \ell \in P_j\}$;
if $m_{ij} > 0$ for some $P_j \in R$
determine the pair $P_j \in R$ such that m_{ij} is maximal;
set $G_{p+1} = P_i \cup P_j$
else
set $G_{p+1} = P_i$
end if
until the complement of $G_1 \cup G_2 \cup \ldots \cup G_{p+1}$ is empty
end. $\square$

Up to now we have only considered variational problems in two dimensions for a scalar equation.

The extension to systems of equations is straightforward. A typical case is the membrane problem. A 2-dimensional displacement vector (u, v) is to be computed for each node of a 2-dimensional grid. Obviously we must not mix the u-variables with the v-variables when we create the coarse grid functions. The system matrix has the form

$$\begin{pmatrix} A_{uu} & A_{uv} \\ A'_{uv} & A_{vv} \end{pmatrix}.$$

The submatrix A_{uu} describes the dependence of the energy on u if alle v-variables are zero. This submatrix describes the geometric structure.

A partitioning of the u-variables will be determined from the submatrix A_{uu} in the way as scalar equations are treated. The partitioning will be translated to the v-variables in an obvious way.

On the other hand, the extension to 3-dimensional grids was considered only for special grids. We have confined ourselves to cases where the 3-dimensional grid consists of identical copies of 2-dimensional layers.

First, a partitioning of one layer into pairs is constructed as in the first stage of the 2-dimensional algorithms. Moreover, a partitioning in the third dimension is constructed by putting pairs of two consecutive layers together. The product of the two partitionings provides the partitioning of the 3-dimensional grid.

3. The Multigrid Algorithm

When problems with 1000 to 20000 unknowns are treated, it is sufficient to perform the coarsening three times. So we have 4 levels. The system on the coarsest level will have 30 to 500 unknowns, and the effort for the Cholesky factorization of the coarsest matrix will be neglegible when compared to the other parts of the computation.

The original grid will be labelled as level 0 while the levels 1, 2 and 3 will refer to the reduced systems of equations.

As was mentioned before, the given system of equations is solved by a conjugate gradient (cg-) iteration, and 1 cycle of the multigrid method at level 0 is taken as a preconditioner. We define recursively the multigrid algorithm AMG_ℓ at level ℓ. For convenience, we restrict ourselves to the V-cycle.

Algorithm $AMG_\ell(b,x)$ for the computation of an MG-approximation of $A_\ell x = b$.
If $\ell = 3$
then
    compute x from the given system $A_\ell x = b$
    by using the Cholesky decomposition of A_ℓ
else
    $x^0 = x$;
    $x^1 = S(A_\ell, b, x^0)$;
    $d = b - A_\ell x^1$;
    evaluate the restriction $\bar{d}$ of d by

$$\bar{d}_j = \sum_{i \in G_j} d_i\,;$$

    apply $AMG_{\ell+1}(\bar{d}, \bar{y})$;
    evaluate the prolongation:

$$y_i = \bar{y}_j \quad \text{for each } i \in G_j;$$

    $x^2 = x^1 + \alpha \cdot y \quad$ with $\alpha = 1.8$;
    $x^3 = S(A_\ell, b, x^2)$;
    $x = x^3$;
end. $\square$

Here $S(A_\ell, b, z)$ denotes the result of one step of the symmetric successive overrelaxation with $\omega = 1$ applied to the vector z. It is applied both before and after the coarse grid correction in order to get an iterative procedure to which a symmetric matrix is associated in a matrix representation. We note that the coarse-grid correction y is multiplied by a factor of 1.8. The reason will be given at the end of this section.

The multigrid procedure has been established on the basis of heuristic arguments. We do not have a complete theory. Nevertheless, we get some insight by looking how the method works on a model problem. Specifically, the choice of the free parameters in the multigrid code is motivated by the answers to the following questions:

1. Does the condition number decrease when we move from the fine grid to the coarser ones?

2. Can the elements from the reduced function spaces provide good approximations of the smooth functions?

We consider the finite element approximation of the Poisson equation on a rectangle

$$\begin{aligned} -\Delta u &= f \quad \text{in } \Omega, \\ u &= 0 \quad \text{on } \partial\Omega. \end{aligned}$$

The discretization by linear finite elements on a rectanguler grid leads to the well-known matrix with the standard 5-point-stencil

$$\begin{pmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{pmatrix}.$$

The associated quadratic form can be expressed as a sum of squares of differences

$$x'Ax = \sum_{i,k}{}' (x_i - x_k)^2.$$

Here, the prime indicates that the sum runs over all pairs (i, k) which belong to neighbours in the rectangular grid.

When the coarsening is constructed and the number of rows and columns are even, it is natural to put the vertices of squares into groups. If we now restrict ourselves to the coarse grid function, i.e., if we assume that

$$x_i = \bar{x}_j \quad \text{for } i \in G_j,$$

then the quadratic form is easily computed:

$$x'Ax = 2\sum_{j,\ell}{}' (\bar{x}_j - \bar{x}_\ell).$$

The number of rows and columns resp. are reduced by a factor of 2. The system matrix for the subspace is a multiple of the matrix which is obtained in classical multigrid codes. In particular, we conclude that the condition number is reduced by a factor of 4.

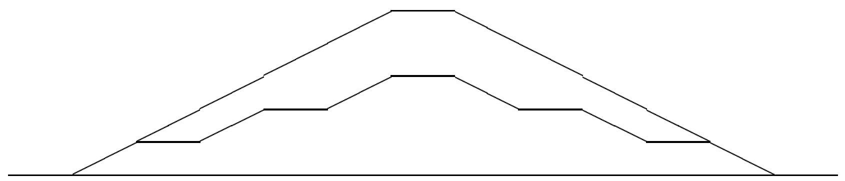

Fig. 3. Approximation of a function which is linear on 3 segments, by a coarse-grid-function

The approximation properties are less satisfactory. This becomes obvious already from the one-dimensional Poisson equation. Fig. 3 shows a "smooth" function which is linear on 3 segments. The set of coarse grid functions consists of those functions which are constant on each second subinterval. The best approximation to the given function in the energy norm is easily characterized. It has the same slope as the given function on those intervals on which the functions are not constant by definition of the subspace. The L_2-distance between the two functions is roughly $\frac{1}{2}$ of the norm of the given function.

The influence of the poor approximation property was reduced by a multiplication of the coarse-grid correction with an overrelaxation factor $\alpha > 1$. The numerical calculations have shown that $\alpha \approx 2.0$ would be optimal for the Poisson equation. This is consistent with the discussion of Fig. 3. On the other hand the dependence on α is not very strong at the optimum, and we prefer smaller values, in order to avoid overshooting. Therefore, $\alpha = 1.8$ was chosen.

4. Details and Variants

In the design of multigrid codes there are several parameters which may be fixed by the designer. Examples of such parameters are the number of smoothing steps, the type of the cycle, the choice of the smoothing operations, and possible overrelaxation factors.

Here, we consider the optimization for using the multigrid cycle as a preconditioner. It has turned out that in this framework the performance is very insensitive to those parameters which are often discussed for standard multigrid procedures. The dominating feature is the fact that our coarse grid corrections sometimes provide only poor approximations of the smooth parts of the error. Indeed, the performance of the iteration is surprisingly better than the quality of the coarse grid correction.

On the finest level, there is only one presmoothing step and one for postsmoothing. If the number of smoothing steps is doubled, then the computing effort on the finest level increases by 66%. This effort is better invested in an augmentation of the number of steps of the outer cg-iteration.

If we increase the number of smoothing iterations on the coarser levels, this has little influence on the computer time per cycle. On the other hand, it has turned out that the influence on the error reduction per cycle is also not significant. Therefore, we have increased the number only after two changes of the level, see Fig. 4.

The same observation was made in connection with the type of the cycle. One may expect that the W-cycle is much better than the V-cycle since the approximation property is not as good as in classical multigrid procedures. We have observed, however, only small differences. Eventually we have chosen the cycle as given in Fig. 4.

More generally, the insensitivity reported above seems to be very typical for this kind of algebraic multigrid. There are only a few exceptions. The overrelaxation discussed at the end of the last section caused a substantial reduction of the iteration steps, c.f. the numerical examples below. Another improvement was successful

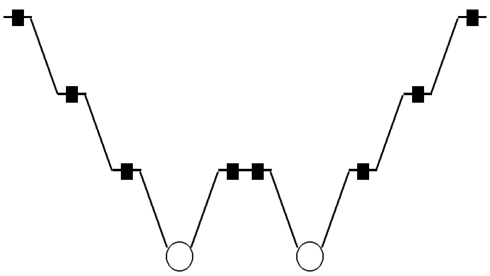

Fig. 4. Resulting multigrid cycle. (-■-) smoothing step, (○) exact solution

in particular with three-dimensional examples. We replaced the point Gauss-Seidel relaxation in the smoothing steps by block Gauss-Seidel relaxation. The blocks were chosen such that we had to invert only tridiagonal matrices.

The reason for the good performance of this procedure is a feature that we find often in 3-dimensional problems. The depth of the domain is often so small that neighbours in the vertical direction are strongly coupled [3]. The block relaxation can cope with this situation while incomplete Cholesky decomposition cannot [12, p. 167], [8]. — On the other hand, the extra effort does not pay if we have nice examples as the model problem with the Poisson equation and the standard 5-point-stencil.

Finally we will comment on some features which are typical for workstations. When we only count the number of operations, then we come to the conclusion that a cg-step with our multigrid preconditioner costs as much as 2.5 steps of a cg-iteration with standard preconditioner. The factor 2.5 may increase up to 3 in some cases, and a step of an incomplete Cholesky decomposition or a step of symmetric SOR relaxation is considered as standard preconditioner.

We must not forget, however, that a multigrid algorithm needs more auxiliary matrices and vectors. Moreover, the prolongations and restrictions use the vectors not in their natural order. Therefore, a much larger amount of data will pass the central processing unit of the computer. If a workstation has only a small memory or a small cache, an iteration step may take much more time than 2.5 steps of a simple cg-code.

5. Numerical Results

We have applied the algorithm to some model problems and to real life problems. The iteration was stopped when the ℓ_2-norm of the residues was reduced by a factor 10^{-5}. In this context, k will always denote the number of iteration steps and t the computer time given in seconds with the time for the set-up included. The time specifications refer to a RISC 6000/320 with 7 MFlops.

It has turned out that the determination of the geometry and the computation of the matrices for the coarse grids costs as much as about 3 cycles of the resulting iteration. Moreover the two methods for the generation of the coarse spaces led to iterations with almost the same performance. Therefore we do not report on the differences.

Table 1. Poisson equation ($n = \ell^2$, k = number of iteration steps for an accuracy of 10^{-5}, t = computation time, n_c = dimension of problem on the coarsest level.)

ℓ	n	ICC		AMG with 4 levels				
					$\alpha = 1.0$		$\alpha = 1.8$	
		k	t	n_c	k	t	k	t
10	100	10	0.01	2	6	0.05	4	0.03
20	400	15	0.08	7	7	0.10	5	0.13
40	1600	26	0.46	25	10	0.48	6	0.38
80	6400	49	3.00	100	12	2.22	7	1.62
120	14400	72	9.84	225	13	5.41	7	3.76

Example 1. The equations which are obtained by the discretization of the Poisson equation on a square with the standard 5-point-stencil have been tested. In Table 1 results for the preconditioning by incomplete Cholesky (ICC) are compared with the algebraic multigrid algorithm (AMG). The latter was performed with and without overrelaxation as specified in Section 3. Obviously the overrelaxation improves the performance substantially.

The results show that the new method is faster than the standard cg-algorithm if $n > 1000$. The difference in computer time is not large.

A second glance at the numbers provides more information. It shows that the number of cycles increases only very slowly with the number of variables.

Table 2. Decrease of the energy norm of the error during the iteration. Poisson equation with $\ell = 120$, $\alpha = 1.8$, AMG preconditioning with 4 levels.

0	$6.87 \cdot 10^3$
1	$1.07 \cdot 10^3$
2	$8.89 \cdot 10^1$
3	$1.14 \cdot 10^1$
4	1.07
5	$1.14 \cdot 10^{-1}$
6	$1.42 \cdot 10^{-2}$
7	$1.20 \cdot 10^{-3}$

Table 2 shows that the error is reduced by a factor of about 10 in each iteration step when measured in the energy norm. So we encounter a performance almost as good as in classical multigrid algorithms.

There is, however, another difference which is important for the treatment of non-linear problems. In our algorithm the reduction of the error is almost uniform. In cg-iterations with classical preconditioners the main reduction of the error is obtained during the last steps. If we need only an error reduction of 10^{-2} insted of 10^{-5} the advantage of the multigrid preconditioner increases.

Example 2. A practical problem with a domain which is very different from a rectangle and with an unstructured grid in $\mathbb{R}^2$ is found in the computation of the magnetic field in a motor. The rotor in a motor has a boundary such that the grid for finite element calculations is an unstructured grid [7]. Moreover there are local refinements. The matrices for this problem and different refinement levels were given to us by U. Langer and his group. We note that they have investigated other algorithms for this real life problem.

Table 3. Example of a motor by Langer et al. (Iteration for an accuracy of 10^{-5}.)

n	ICC		AMG	
	k	t	k	t
410	23	0.09	9	0.16
1580	66	1.06	11	0.68
6203	137	8.01	12	3.06
10480	158	16.09	14	6.05

The numerical results in Table 3 show that the convergence of our algorithm is only slightly slower than in the model example with the rectangular domain above. The number of iteration steps increases very slowly, also in this case. [When we compare these results with those for other examples below we may conclude that a very good grid generator was used here and that the condition numbers of the matrices are consistent with the problem.]

Example 3. The Lamé equations for the membrane problem in 2-dimensional elasticity

$$\mu \triangle u + (\lambda + \mu)\,\mathrm{grad}\,\mathrm{div}\, u = f$$

are an example of a system of 2 coupled equations of second order. Numerical calculations were done for rectangular membranes. Specifically, we assumed that Neumann boundary conditions are given on two sides and that we have a material with Poisson ratio $\nu = 1/3$, i.e., $\lambda = 2\mu$.

The numerical results in tables 4 and 5 show that the error reduction factor per step is about 0.35. The convergence is slower than for the Poisson equation. It is interesting to note that classical multigrid algorithms are also slower here [6].

The same is reported for the algebraic multigrid procedure of Ruge and Stüben [10]. Since cg-iterations with standard preconditioners are also slower, the preconditioning with algebraic multigrid is advantageous for medium size problems.

Table 4. Membrane problem with $\nu = 1/3$. (ℓ = length and w = width of membrane in multiples of h. Iteration for an accuracy of 10^{-5}, n_c = dimension of problem on the coarsest level.)

$\ell * w$	n	ICC		AMG with 4 levels				
					$\alpha = 1.0$		$\alpha = 1.8$	
		k	t	n_c	k	t	k	t
$10 * 20$	462	31	0.22	8	11	0.20	9	0.15
$20 * 40$	1722	59	1.52	28	15	0.86	10	0.67
$40 * 80$	6642	114	10.20	100	17	3.91	12	3.00
$60 * 120$	14762	170	33.90	243	19	9.85	13	7.30

Table 5. Decrease of the energy norm of the error during the iteration. Membrane problem with $\ell * w = 60 * 120$, $\alpha = 1.8$, AMG preconditioning with 4 levels.

0	2.38
1	1.17
2	$4.76 \cdot 10^{-1}$
3	$1.84 \cdot 10^{-1}$
4	$6.06 \cdot 10^{-2}$
5	$1.92 \cdot 10^{-2}$
6	$7.37 \cdot 10^{-3}$
7	$2.31 \cdot 10^{-3}$
8	$7.01 \cdot 10^{-4}$
9	$2.35 \cdot 10^{-4}$
10	$8.91 \cdot 10^{-5}$

Examples 4–6. Linear equations which arise in typical 3-dimensional groundwater flow simulations are often so ill-conditioned that cg-iterations with standard preconditioners fail, c.f. [3]. (Specifically, the equations in example 6 were solved in the actual computations by a complete Cholesky decomposition which took 1 hour while a cg-iteration with adapted preconditioners can be done in less than 2 minutes.)

In groundwater calculations the 2-dimensional projections of the domains are often partitioned such that triangles with very small angles arise. Specifically, there are many angles less than 5^0 in the meshes of the Examples 5 and 6, and this implies that the cg-iteration with preconditioning by incomplete Cholesky decompositon does not work well. Table 6 shows that the AMG-preconditioner reduces the number of iteration steps substantially. [We note that the SSOR relaxation is performed with $\omega = 1.0$ when it is used as a smoother for AMG while $\omega = 1.3$ is chosen when it serves as a preconditioner.]

Table 6. Three examples of matrices in groundwater simulations. (Iteration for an accuracy of 10^{-5}.)

	Example 4		Example 5		Example 6	
variables	9246		16464		45666	
number of hor. layers	6		6		12	
	k	t	k	t	k	t
preconditioner						
ICC	49	7.8	$--$		$--$	
point SSOR	183	24.1	620	150	750	280
line SSOR	30	4.6	278	83	150	60
AMG with line SSOR	7	5.4	128	105	30	81

On the other hand on workstations the computer time was larger than for the computation with block-SSOR preconditioners even in the case when the number of iteration steps was reduced by a factor of 5. The corresponding factor for the number of floating point operations is about 2. The computing time, however, turned out to be larger due to the small cache in workstations of medium size. In these examples the number of nonzero entries in the matrices were $\approx 9n$.

6. Conclusion

An algebraic multigrid method with simple prolongation and restriction operators has been presented. The performance is better than one expects when seeing the simple operators. Good error reduction factors were found in real life problems if good mesh generators were used in the discretization process.

References

[1] R.E. Bank: An algorithm for coarsening unstructured meshes. (submitted)

[2] D. Braess, M. Biebighäuser, P. Grassberger and R. Leuverink: Multi-grid methods for steady state diffusion in random media. J. Comput. Physics 107, 118–123 (1993).

[3] D. Braess and Ch. König: A preconditioning technique for the fast solution of 3-D groundwater problems (in preparation).

[4] F. Chatelin and W.L. Miranker: Acceleration by aggregation of successive approximation methods. LAA 43, 17–47 (1982).

[5] W. Hackbusch: Multi-Grid Methods and Applications. Springer-Verlag Berlin, Heidelberg 1985.

[6] M. Jung: Konvergenzfaktoren von Mehrgitterverfahren für Probleme der ebenen, linearen Elastizitätstheorie. ZAMM 67, 165–173 (1987).

[7] M. Jung and U. Langer: Applications of multilevel methods to practical problems. Surv. Math. Ind. 1, 217–257 (1991).

[8] R. Kettler and P. Wesseling: Aspects of multigrid methods for problems in three dimensions. Appl. Math. and Comp. 19, 159–168 (1986).

[9] J.W. Ruge and K. Stüben: Efficient solution of finite difference and finite element equations by algebraic multigrid (AMG). In "Multigrid Methods for Integral and Differential Equations" (Paddon, D.J.; Holstein, H., eds.), pp. 169–212, Clarendon Press, Oxford 1985.

[10] J.W. Ruge and K. Stüben: Algebraic multigrid (AMG). In "Multigrid Methods" (St. Mc Cormick, ed.), Frontiers in Applied Mathematics, Vol 5, SIAM, Philadelphia 1986.

[11] K. Stüben: Algebraic multigrid (AMG): Experiences and comparisons. Appl. Math. Comput. 13, 419–452 (1983).

[12] P. Wesseling: An Introduction to Multigrid Methods. John Wiley & Sons, Chichester – New York 1992.

[13] P.M. de Zeeuw: Matrix-dependent prolongations and restrictions in a blackbox multigrid solver. J. Comp. and Appl. Math. 33, 1–27 (1990).

Dietrich Braess
Institut für Mathematik
Ruhr-Universität Bochum
D-44780 Bochum

braess@num.ruhr-uni-bochum.de