APPLIED MATHEMATICS AND COMPUTATION 13:311-326(1983)

0096-3003/83/$03.00

MULTIGRID METHODS: DEVELOPMENT OF FAST SOLVERS

P.W. Hemker*), R. Kettler **)***), P. Wesseling **) and P.M. de Zeeuw *)

*) Mathematical Center, Amsterdam, The Netherlands.
**) Delft University of Technology, Department of Mathematics and Informatics, Julianalaan 132, 2628 BL Delft, The Netherlands.
***) Kon. Shell Expl. Prod. Lab., Rijswijk, The Netherlands.

Paper presented at the International Multigrid Conference, April 6-8, 1983, Copper Mountain, Colorado, U.S.A.

ABSTRACT

Numerical and programming aspects are discussed of multigrid algorithms for the solution of discretized linear elliptic equations. The aim is to obtain software that is perceived and can be used just like any standard subroutine for solving systems of linear equations. The user has to specify only the matrix and the right-hand-side, and remains unaware of the underlying multigrid method. We find that a large class of equations can be solved efficiently in this way. The equation may be non-self-adjoint, and its coefficients are arbitrary. Special attention is given to the treatment of the convection-diffusion equation at high Péclet number. Details are given of an available portable FORTRAN code, which vectorizes satisfactorily on vector machines. CP time statistics are given for a CYBER-205.

1. INTRODUCTION

In this paper the multigrid approach is studied with the aim of developing algorithms for the solution of discretized linear elliptic equations with general coefficients. These algorithms should be efficient for a large class of problems, so that the user seldom needs to make decisions as to what to use when. Also, the algorithm should be perceived by the user just like any standard subroutine for solving linear systems of equations. The user has only to give the matrix and the right-hand-side in a prescribed data-structure, and remains unaware of the underlying multigrid algorithm. The algorithm should be presented in a user-friendly portable computer code.

We will discuss how the main ingredients of the multigrid methodology can be chosen such that the above objectives are reached. Details are given of a FORTRAN code called MGD1V; see also [18, 19]. MGD1V is an example of what has been called "black box" multigrid method in [5]; there another example of such a method is presented, which is particularly suited for problems with discontinuous coefficients. A similar method has been developed in [9, 10, 11].

MGD1V has been tested on sequential and vector machines. It auto-vectorizes to a satisfactory degree. CP time statistics will be given. For the availability of MGD1V, see the note at the end of this paper.

Of course, the multigrid approach can be and is used for much wider purposes, for example to solve nonlinear differential equations in novel ways, to develop new adaptive discretizations, and to intertwine discretization and solution. For a review of the present and future potential of the multigrid approach, see [2, 3, 4].

2. THE PROBLEM TO BE SOLVED

The differential equation to be solved is in Cartesian subscript notation given by:

$$-(a_{ij}u_{,j})_{,i} + (b_i u)_{,i} + cu = f \tag{2.1}$$

in $\Omega = (0,1) \times (0,1)$. If Ω is not square, it can be mapped on $(0,1) \times (0,1)$ numerically, using for example the boundary fitted coordinate approach. This mapping can be carried out efficiently with the multigrid software to be discussed. The boundary conditions may be of Dirichlet, Neumann or mixed type. Periodic boundary conditions have not yet been implemented, but will be in the near future. The coefficients are arbitrary, but satisfy the ellipticity condition:

$$a_{ij}\xi_i\xi_j > C\xi_i\xi_i \quad , \qquad \forall\, \xi_i \in \mathrm{IR} \, , \quad i = 1,2. \tag{2.2}$$

The problem is discretized with the 7-point or 9-point difference molecules depicted in Fig. 2.1.

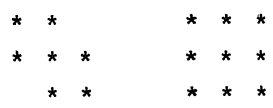

Fig. 2.1 7- and 9-point molecules

An equidistant computational grid Ω^ℓ is defined by:

$$\Omega^\ell \equiv \{(x_1,x_2) \mid x_i = m_i 2^{-\ell}, \; m_i = 0(1)2^\ell\}. \tag{2.3}$$

If the user wants a non-uniform mesh he has to use a mapping. The mesh-sizes must be negative powers of 2. A suitable 7-point difference approximation of (2.1) is:

$$-\frac{1}{2}(\nabla_i a_{ij}\Delta_j + \Delta_i a_{ij}\nabla_j)u^\ell + \delta_i(b_i u^\ell) + cu^\ell = f^\ell \, , \tag{2.4}$$

where the superscript ℓ denotes grid functions on Ω^ℓ, ∇_i and Δ_i are backward and forward difference operators in the x_i-direction, and δ_i is a suitable difference operator. For example, when choosing upwind differences, $\delta_1(b_1 u^\ell)$ can be given by:

$$\begin{aligned}\delta_1(b_1u^\ell)_{ij} = \tfrac{1}{2}\,2^{-\ell}\{&-(b_{1,i-1,j}+|b_{1,i-1,j}|)u^\ell_{i-1,j} + 2|b_{1,ij}|u^\ell_{ij} \\ &+ (b_{1,i+1,j}-|b_{1,i+1,j}|)u^\ell_{i+1,j}\}.\end{aligned} \tag{2.5}$$

A non-conservative formulation and/or a 9-point discretization can be used as well, without consequences for the rate of convergence of multigrid methods. Another possibility is to choose a central difference approximation for δ_i. Then our multigrid methods diverge for a_{ii} small enough. An artificial viscosity scheme results if one adds an artificial viscosity coefficient $\nu_h\delta_{ij}$ to a_{ij}, while keeping δ_i central. The accuracy of the resulting scheme is comparable to that of the upwind scheme, but the convergence behaviour of multigrid methods can be different for the two schemes, as we shall see. If no mixed derivative is present, we have the familiar 5-point difference scheme.

3. TESTPROBLEMS

It is hard to obtain a watertight guarantee that a given method works for a large class of problems. Therefore multigrid practitioners have adopted test problems that are representative of the difficulties that may be encountered in practice. A rather general set of test problems is:

(i) The convection-diffusion equation:

$$\cos \alpha \, u_{,1} + \sin \alpha \, u_{,2} = \varepsilon \, u_{,ii} + f. \tag{3.1}$$

(ii) The anisotropic diffusion equation:

$$\varepsilon \, u_{,11} + u_{,22} = f. \tag{3.2}$$

(iii) Equations with a mixed derivative:

$$u_{,11} + 1.7 \, u_{,12} + u_{,22} = f, \tag{3.3}$$

$$u_{,11} - 1.7 \, u_{,12} + u_{,22} = f. \tag{3.4}$$

Our ideal is to obtain a multigrid method that works for all equations of the type (2.1). An essential limitation of numerical experiments on test problems is, that a general assertion, such as: this method works for all equations of type (2.1), cannot be proved by means of examples. This cannot be helped. At first sight, (3.1) - (3.4) seem to be rather special, because the coefficients are constant. However, this fact is not exploited. Moreover, these constant coefficient problems are often more difficult than equations with varying coefficients, because it may easily happen, that the performance of a method is bad for certain values of α or ε. In problems where α and ε vary widely in Ω this fact goes more easily unnoticed than in a test problem where α and ε have a constant unfavorable value throughout Ω.

An important limitation of constant coefficient test problems is, that the results obtained are not representative for problems with discontinuous coefficients. Suitable test problems for this situation have been solved with multigrid methods in [1, 5, 9, 10, 11].

We will subject various multigrid methods to the three test problem, for a wide range of values $\alpha \in [0,2\pi)$ and $\varepsilon \in (0,\infty)$. In some cases the ordering of the grid point and unknowns influences the results. We always take the x_1- and x_2-axis horizontal and vertical in the usual way, and order grid points and unknowns on an N×M grid as follows:

```
NM-M+1 . . . . . NM
  .                .
  .                .
 M+1    M+2  . . . 2M
  1      2   . . . M
```

The test problems can be used in two ways. The first is to analyse a method theoretically, using Fourier methods. An introduction to this type of analysis is given in [14], together with a large collection of results. The second possibility is to run the multigrid method and see what happens. Advantages of Fourier analysis are, that one may obtain the spectral radius and even norms of the iteration matrix, and that one can sometimes understand the effect of modifications of the methods theoretically. Limitations are, that the analysis is restricted to two-grid methods, and that in many cases the presence of boundaries is neglected. For the V and sawtooth (see below) cycles two-grid analysis is not quite representative of multigrid performance. For methods using non-identical discretizations on the various

grids, as is often the case with Galerkin coarse grid approximation, two-grid analysis does not detect possible shortcomings on coarser grids. For problems with strong coupling in a certain direction, such as with strongly anisotropic diffusion or with strong convection, the influence of the boundary often extends over all Ω, even for a fine mesh. Nevertheless, Fourier analysis is a valuable research tool, if one keeps the aforementioned limitations in mind.

Here we do not report results of Fourier analysis, but restrict ourselves to numerical experiments with the full algorithms. Where our results deviate from Fourier analysis predictions catalogued in [14], the limitations mentioned above play a role.

4. PROLONGATIONS AND RESTRICTIONS

A set of coarse grids $\{\Omega^k,\ k = 1(1)\ell-1)\}$ is defined by eq. (2.3) with ℓ replaced by k. The spaces of grid-functions on Ω^k are called U^k. Prolongation and restriction operators are denoted by p^k and r^k:

$$p^k : U^{k-1} \to U^k , \quad r^k : U^k \to U^{k-1} \tag{4.1}$$

Often, these operators are denoted respectively by I_{k-1}^k and I_k^{k-1} in the literature; the above notation saves one index. Prolongation is often also called interpolation, and restriction weighting (in special cases injection, or half-weighting, or full-weighting). These operators are practically identical to what is called prolongation and restriction, and denoted by p and r, in other fields of mathematics, see for example [15]. Sometimes p^k is not easily recognized as resulting from interpolation.

One can choose p^k and r^k in many ways. One possibility for p^k is linear interpolation. We will use interpolation in triangles; for a precise definition (7-point prolongation) see [18]. This will be referred to as P7 in the sequel.

Furthermore, we will consider two prolongations that have been developed for cases where a_{ij} is discontinuous; not because discontinuous coefficients are considered here, but because these prolongations work well for (3.1). One is the first of the two prolongations used in [1], denoted here by P9'. (The second, also used in [5], gives similar results as the first for the test problems discussed here, and will not be considered.) Another, similar, prolongation is used in [10, 11], and will be denoted here by P9". The difference between P9' and P9" is, that P9" uses the right hand side.

In all cases, the restriction is taken to be the adjoint of the prolongation, disregarding the use of the right hand side for P9".

5. COARSE GRID APPROXIMATION

Let the problem to be solved including suitable boundary conditions be denoted in matrix notation as follows:

$$A^\ell u^\ell = f^\ell \tag{5.1}$$

On the coarse grids Ω^k, $k = \ell-1(-1)1$ we need so-called coarse grid operators A^k, $k = \ell-1(-1)1$, that approximate A^ℓ and each other. Here the requirement that the user has to provide only A^ℓ and f^ℓ plays a crucial role. This requirement implies, that the program must generate automatically accurate operators A^k, $k = \ell-1(-1)1$. This is a prime motivation to use Galerkin coarse grid approximation:

$$A^{k-1} = r^k A^k p^k , \quad k = \ell(-1)2. \tag{5.2}$$

Generally, the resulting A^k are at least as accurate and dependable as user-generated finite difference coarse grid operators. For some comparative experiments, see [18, 19]. Some easily accessible publications in which Galerkin coarse grid approximation is used are [1, 5, 6, 10, 11, 16, 17]. For a justification of the appellation "Galerkin coarse grid approximation", and for a few remarks on efficient programming of $r^k A^k p^k$, see [18] and [19] respectively.

6. SMOOTHING PROCESSES

We exclude smoothing processes for which it is already known that they do not work well for (3.1)-(3.4), for arbitrary ε and α. Looking at the extensive catalogue of smoothing analysis in [10], the following smoothing processes have been selected for inclusion in the numerical experiments to be described in the present paper: ILU (1,2) (to be denoted as ILU) and ILLU. Furthermore, because of the favourable results obtained in [14] for self-adjoint problems, we have also included ZEBRA relaxation with horizontal lines, to be denoted by HZ, and alternating ZEBRA relaxation (taking both horizontal and vertical lines), to be denoted by AZ. We take first odd lines, then even lines; the boundary lines are odd. With first even, then odd lines, lower rates of convergence were obtained. With AZ we take first horizontal, then vertical lines. For a precise definition of ILU, ILLU, HZ and AZ, see [18, 19, 10, 14]. ILU and ILLU are examples of iterative methods based on incomplete LU-decompositions. These were first introduced in [12] as preconditionings for conjugate gradient methods. For applications to multigrid methods and for smoothing analysis of incomplete LU smoothing processes, see [7, 8, 10, 11, 16, 18, 19, 20]. For easy-to-program formulae, see [19].

7. MULTIGRID CYCLES

The class of multigrid methods to be considered can be denoted in quasi-Algol notation as follows (cf. [18]):

```
Procedure multigrid method (k), value k, integer k;
begin integer n;
    if k = kcoarse then Sa(k,u^k,A^k,f^k) else
    begin for n:=1(1) sa [k] do Sa(k,u^k,A^k,f^k);
        f^(k-1):=r^k(f^k-A^k u^k); u^(k-1):=0;
        for n:=1(1) sc [k] do multigrid method (k-1);
        u^k:=u^k+p^k u^(k-1);
        for n:=1(1) sb [k] do Sb(k,u^k,A^k,f^k)
    end
end multigrid method;
```

If kcoarse=1 then the coarsest grid that is used consists of 3×3 points: for kcoarse=2 it has 5×5 points, etc.; cf. (2.3). Sa and Sb are smoothing processes, which may depend on k. For example, for k = kcoarse Sa is often

an exact solution method. For sc[k]=2, $\forall k$, we have the so-called W-cycle; for sc[k]=1, $\forall k$, we have the V-cycle; cf. [14]. If sc[k]=1 and either sa[k]=0 or sb[k]=0, $\forall k$, we have a special case of the V-cycle, called the sawtooth cycle. Generally speaking, none of these cycles is found to be significantly more efficient than the others for the test problems and multigrid options that we have considered; with a good smoothing process and good coarse grid approximations the efficiency of the various cycles is usually roughly the same (see [8, 18]). Another type of method, not represented by the above procedure, is the so-called full multigrid method; for a description see [4]. We have not experimented with this method. It would probably require more interaction of the user with the algorithm than programs based on the above procedure. In order not to burden the user with the choice of parameters and to make the algorithm autonomous, i.e. independent of the user, a fixed choice is made for sa, sb, sc, Sa, Sb and A^k. For example, in the program MGD1 ([19]) sa=0, sb=1, sc=1, Sa and Sb are ILU, prolongation and restriction are of P7 type, and A^k is given by (5.2). This strategy has been implemented in the FORTRAN program described in the following section. For non-recursive programming of the W-cycle, see for example [14].

8. AN AUTONOMOUS MULTIGRID FORTRAN PROGRAM

The following is an outline in quasi-FORTRAN of the multigrid algorithm MGD1.

```
C     MULTIGRID PROGRAM MGD1
C     INITIAL GUESS IS u^ℓ=0
      DO 10 k=ℓ-1(-1)kc
      CALL RESTRICTION (f,f,k)              f^k = r^{k+1} f^{k+1}
   10 CONTINUE
C     START OF maxit MULTIGRID ITERATIONS
      DO 50 n=1, maxit
      IF (n.EQ.1) GO TO 30
      CALL CTUMV (C,u,v)                    v^ℓ = C^ℓ(u^ℓ-v^ℓ)
C     v^ℓ IS THE NEW RESIDUE f^ℓ-A^ℓ u^ℓ
      CALL RESTRICTION (f,v,ℓ-1)            f^{ℓ-1} = r^ℓ v^ℓ
      DO 20 k=ℓ-2(-1)kc
      CALL RESTRICTION (f,f,k)              f^k = r^{k+1} f^{k+1}
   20 CONTINUE
   30 CALL SOLVE (u,f,kc)                   u^1 = (L^1 U^1)^{-1} f^1
      DO 40 k=kc+1(1)ℓ-1
      CALL PROLONGATION (u,u,k)             u^k = p^k u^{k-1}
      CALL CTUPF (v,u,f,k)                  v^k = C^k u^k + f^k
      CALL SOLVE (u,v,k)                    u^k = (L^k U^k)^{-1} v^k
   40 CONTINUE
      CALL PROLONGATION (v,u,ℓ)             v^ℓ = p^ℓ u^{k-1}
      v^ℓ=v^ℓ+u^ℓ
      CALL CTUPF (u,v,f,ℓ)                  u^ℓ = C^ℓ v^ℓ + f^ℓ
```

```
      CALL SOLVE (u,u,ℓ)                    u^ℓ = (L^ℓ U^ℓ)^-1 u^ℓ
   50 CONTINUE
```

A portable FORTRAN code for the MGD1 algorithm called MGD1V has been implemented. Prolongation and restriction are of P7 type (section 4), and smoothing is done with ILU (section 6). The matrix C is defined by C = LU-A, with L and U the incomplete LU-decomposition factors. For ILU, C has only two non-zero diagonals, and is used for cheap residue calculation. More details can be found in [18, 19]. In MGD1V, C is not stored but computed from L and U at the expense of one (vectorizable) multiplication per element.

MGD1V has been designed for auto-vectorization on vector computers, such as the CYBER-205 and the CRAY-1, without having to sacrifice anything on sequential machines. A version especially suited for the CYBER-205 is called MGD1D. Compared with straightforward FORTRAN programming, particularly the construction of L^k, U^k and subroutine SOLVE are implemented differently, although in a straightforward way. The main idea is to split off vectorizable loops as much as possible by referring to individual grid-lines. In SOLVE a simple recursion is left, which in the case of MGD1D is speeded up by using calls to the STACKLIB library. These particularly efficient FORTRAN subroutines are provided with the CYBER 205 for elementary algebraic operations that are not vectorizable because of recursion. Some CP time statistics for MGD1D obtained on a CYBER-205 are listed below, comparing with an efficient scalar version. On a 257×257 grid the total speed-up ratio was

10 iterations	sequential	vectorized	ratio
MGD1D:total time	2.141	0.469	4.6
RESTRICTION	0.054	0.033	1.6
CTUMV	0.315	0.010	31.5
SOLVE	0.594	0.263	2.3
PROLONGATION	0.093	0.023	4.0
CTUPF	0.390	0.014	27.9
Preliminary computation of:			
A^k, k=ℓ-1(-1)2	0.317	0.054	5.9
L^k, U^k, k=2(1)ℓ	0.195	0.043	4.5

Table 7.1 CP time statistics for MGD1D on a CYBER 205. Seconds. Total time spent in various subroutines for the execution of 10 iterations, and preliminary computations. Finest grid 129×129, coarsest 5×5.

found to be 5.6.
The same problem takes about 25 seconds on an Amdahl V6.

As we will see shortly, the residue is typically reduced by a factor 0.1 during one iteration, so that one typically obtains a residue reduction of 10^5 in 0.27 seconds on the CYBER-205, for a general elliptic differential equation with continuous coefficients on a 129×129 equidistant grid. An exception has to be made for certain convection-diffusion cases, as we will see, but the preceding statement remains valid for 65×65 grids (or smaller) at roughly 1/4 of the time mentioned (or less). Other versions of MGD1, using ILLU and/or the P9' or P9" prolongations and restrictions, for better

performance in the convection diffusion case, for discontinuous coefficients and making treatment of arbitrary regions easy, will be brought out in the near future.

9. RESULTS FOR THE CONVECTION-DIFFUSION TEST PROBLEM

The following table gives some observed average reduction factors for MGD1V for the convection-diffusion testproblem (eq. (3.1)). Here f=1.

$\varepsilon=10^{-8}$, Nf=65, Nc=3		
α	Ni	ρ
0	4	$.5_{10}{}^{-5}$
15	3	$.4_{10}{}^{-6}$
30	3	$.1_{10}{}^{-6}$
45	3	$.1_{10}{}^{-6}$
60	3	$.8_{10}{}^{-7}$
75	3	$.4_{10}{}^{-7}$
90	1	$.2_{10}{}^{-10}$
105	10	0.05
120	10	0.07
135	10	0.13
150	10	0.29
165	10	0.43

$\alpha=165$, Nf=129, Nc=3		
ε	Ni	
10^{-8}	-	div
10^{-5}	-	div
10^{-3}	10	0.41
10^{-2}	10	0.08
10^{-1}	10	0.08
1	10	0.07
10	10	0.07

$\alpha=165$, Nf=65, Nc=3		
ε	Ni	ρ
10^{-6}	10	0.43
10^{-4}	10	0.47
10^{-2}	10	0.09
1	10	0.07

Table 9.1 Results for eq. (3.1) with MGD1V. Finest grid: Nf×Nf; coarsest grid: Nc×Nc; Ni: number of iterations; ρ: average reduction factor Euclidean norm of residue over first Ni iterations.

The initial guess is zero. The value zero is prescribed at the boundaries; this Dirichlet boundary condition is eliminated. We have restricted α (degrees) to the first two quadrants, because the behaviour in the third and fourth quadrants is symmetric with the behaviour in the first two.

It is found that MGD1V performs well for arbitrary ε and α as long as $\ell<6$ (Nf $<$ 65). For $\alpha \in (0^0, 90^0)$ ILU is almost exact for small ε. This is suggested by table 9.1, and confirmed by the smoothing analysis results in [10], which show that not only short but also long wavelength error components are strongly damped. One smoothing on the finest grid almost produces the solution, and what happens on the coarser grids does not matter very much. In the second quadrant the long wavelength error components are not strongly damped, and good coarse grid corrections are necessary to obtain good convergence. However, with each application of (5.2) the difference operator A^k becomes more symmetric and the main diagonal becomes weaker; see some examples of coarse grid difference molecules generated with

(5.2) in [19]. As a result the smoothing properties of ILU deteriorate, and "wiggles" may occur on the coarser grids. The situation gets progressively worse as the number of grids increases, until divergence occurs with 7 levels for α in a fairly narrow band around 165^0, where ILU happens to be the least accurate. For other values of α, ILU is sufficiently accurate to correct the bad approximations generated on the coarse grids. This is dependent on the use of upwind differences on the finest grid. With artificial viscosity discretization, ILU comes less close to being exact, and divergence can also occur on a 65×65 grid.

As already noted, for Nf ≤ 65 MGD1V is dependable and efficient, and because a 65×65 grid is sufficiently fine in many applications, we consider MGD1V a useful working tool, taking into account that it also works for the test problems still to be considered. However, it is desirable to make it work on very large grids too. After all, for very large grids multigrid may be the only viable way to keep the computation time within reasonable bounds.

One way to get MGD1 working on very large grids for convection-diffusion problems with small ε is to replace ILU by a smoothing process which reduces both short and long wavelength components for all directions α sufficiently to compensate for the bad coarse grid corrections. The smoothing analysis results in [10] give hope that ILLU has the desired properties.

The following table gives a typical result.

α	ε	Nf	Nc	ρ
135	$0.31_{10}{}^{-7}$	129	3	$0.14_{10}{}^{-5}$

Table 9.2 Average reduction factor for ILLU smoothing.

Here f=0, and zero Dirichlet boundary conditions are prescribed, so that the solution is identically zero. The initial guess consists of random numbers. In order to illustrate the robustness of the algorithm, now we do not eliminate the Dirichlet boundary conditions, but keep them in the finest grid matrix with a weighting factor 10^p. We choose p large, namely p=10, so that we obtain the same good rate of convergence as when the boundary conditions are eliminated, but not too large, to avoid overflow. Under these conditions the value of p has little influence. We have also applied a homogeneous Neumann condition at outflow boundaries (results not reported here), again, we found little difference. The foregoing remarks apply to all further experiments to be described.

In table 9.2, the value of α is for the worst case that we encountered; for other values of α that we tried convergence is faster. The reduction factor ρ is averaged over the second and third iteration; from now on we exclude the first iteration because it often accidentally gives an untypically small reduction factor. Apparently, ILLU is almost exact for small ε for every α, using upwind differences.

Here we could end our discussion of the convection-diffusion test problem. However, it is not esthetically pleasing to use a multigrid method under circumstances where the coarse grid corrections merely have an adverse effect. We have therefore developed better coarse grid approximations. An analysis of multigrid methods for convection-diffusion problems has been made in [20] and [21]. There a suitable stability concept is introduced, to be denoted here for brevity as S-stability. It is found that with S-stability reasonable rates of convergence are obtained, also with smoothing processes less formidable than ILLU. Methods using P7 and Galerkin coarse grid approximation are found to be not S-stable. They can be made S-

stable by adding artificial viscosity to the coarse grid operators. By choosing a suitable amount of aritificial viscosity satisfactory rates of convergence are obtained. However, adding suitable amounts of artificial viscosity in a user-independent way is somewhat complicated. We therefore stick to (5.2) without adding artificial viscosity, but introduce some "upwind" effect by changing the prolongation and the restriction. It turns out that the P9' and P9" prolongations and restrictions (see section 4) do the job. The following table gives the final ($\ell \to \infty$, $k = 1$) molecule produced by (5.2), using P7, P9' and P9", for a few finest grid molecules. Scaling factors have been disregarded.

	P7	P9', P9"
$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$	$\begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix}$	$\begin{matrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{matrix}$
$\begin{matrix} 0 & 0 & 0 \\ -a & a+b & 0 \\ 0 & -b & 0 \end{matrix}$	$\begin{matrix} 0 & -a+2b & a+b \\ -2a+b & 0 & 2a-b \\ -a-b & a-2b & 0 \end{matrix}$	$\begin{matrix} 0 & 0 & 0 \\ -2a & 2a+2b & 0 \\ 0 & -2b & 0 \end{matrix}$
$\begin{matrix} 0 & -b & 0 \\ -a & a+b & 0 \\ 0 & 0 & 0 \end{matrix}$	$\begin{matrix} 0 & -a-2b & a-b \\ -2a-b & 0 & 2a+b \\ -a+b & a+2b & 0 \end{matrix}$	$\begin{matrix} 0 & -2b & 0 \\ -2a & 2a+2b & 0 \\ 0 & 0 & 0 \end{matrix}$
Finest grid molecule	Final molecule	

Table 9.3 Some difference molecules resulting from coarse grid Galerkin approximation.

In order to illustrate the quality of the coarse grid approximations generated by (5.2) with P9' or P9", we have solved (3.1) with a bad smoothing process, namely line-Gauss-Seidel with the wrong ordering of the lines (LGSB1, see [10]).

The results are given in the following tables.

$\varepsilon\backslash\alpha$	0	15	45	90	135	165
.31	.14	.15	.16	.17	.17	.16
.031	.11	.18	.24	.26	.23	.19
.0031	.06	.36	div	div	div	.42
.00031	.002	.43	div	div	div	.43

Table 9.4 Average reduction factors, P7 prolongation and restriction.

$\varepsilon\backslash\alpha$	0	15	45	90	0	15	45	90
.31	.12	.13	.15	.16	.09	.10	.11	.12
.031	.11	.17	.27	.31	.09	.13	.22	.26
.0031	.06	.39	.66	.77	.05	.34	.59	.68
.00031	.002	.43	.81	.94	.001	.37	.66	.89
	P9'				P9*			

Table 9.5 Average reduction factors.

Because of symmetry, only the first two quadrants have to be considered in table 9.4, and only the first quadrant in table 9.5.

Tables 9.4 and 9.5 give the reduction factor of the maximum norm of the error over iterations 2-10. In order to avoid rounding error effects rescaling is applied. The V-cycle is used (sc[k]=sa[k]=sb[k]=1 in the first algorithm of section 7). The finest grid is of dimension 33×33, the coarsest 3×3. Tables 9.4 and 9.5 clearly show that the bad smoothing process used is not saved by P7, but saved a great deal by P9' and P9", which provide more effective coarse grid corrections. Because P9" is found to be slightly better than P9' (this was found in other cases too), we will report results for P9" only from now on for the convection-diffusion problem.

We now combine P9" with better smoothers: ILU, AZ and ILLU (see section 6). The following results are obtained, for the test problem treated in tables 9.4 and 9.5.

	ILU					
ε\α	0	15	45	90	135	165
.31	.06	.06	.06	.06	.06	.06
.031	.05	.04	.04	.06	.05	.05
.0031	.08	.04	.02	.01	.15	.15
.00031	.02	.007	.002	.0003	.18	.24

	AZ				ILLU			
ε\α	0	15	45	90	0	15	45	90
.31	.07	.07	.07	.07	.04	.04	.04	.04
.031	.06	.07	.06	.05	.04	.04	.03	.02
.0031	.09	.07	.10	.09	.01	.02	.009	.002
.00031	.03	.06	.09	.03	.0002	.002	.001	.0001

Table 9.6 Average reduction factors

For table 9.6 the situation is as described for tables 9.4 and 9.5, except that the V-cycle has been replaced by the sawtooth cycle (sc[k]=1, sa[k]=0, sb[k]=1 in the first algorithm of section 7). The finest grid has dimension 33×33, the coarsest 17×17; we found these 2-level results to be representative for the corresponding 5-level results (with a 3×3 coarsest grid). On the 17×17 grid the equation is solved almost exactly with a sufficient number of relaxations. The following table gives some 5-level results (finest grid 33×33, coarsest 3×3), comparing the sawtooth and the V-cycle. The smoothing process is ILLU.

	sawtooth cycle				V-cycle			
ε\α	0	15	45	90	0	15	45	90
.31	.05	.04	.04	.04	.009	.009	.009	.007
.031	.04	.03	.03	.02	.01	.009	.005	.006
.0031	.01	.02	.009	.002	.0001	.003	.003	.0000
.00031	.0002	.002	.001	.000	.0000	.0000	.0000	.0000

Table 9.7 Average reduction factors.

In the V-cycle twice as much smoothing takes place as in the sawtooth cycle, so that it is about twice as expensive. Therefore table 9.7 suggests that sawtooth is more efficient than V in the present case.

For a comparison of the efficiency of the various methods the decisive factor is the cost of the smoothing process. This depends on the number of atoms in the difference molecule. We consider only the finest grid, because there the bulk of the work takes place. Usually, we there have a 5-point molecule. Giving operation counts is a little tricky, because one can exchange work for storage to a certain extent, and also often save a few operations by clever programming. In [14], table 9.6b and [10], appendix C almost the same operations counts are found for AZ and ILU, assuming that LGSF1F2 in [10] has the same cost as AZ; in [10] it is found that ILLU is about 1.5 times as expensive.

From the experimental results described above we draw the following conclusions for convection-diffusion problems.

For grids not larger than 65×65 the FORTRAN code MGD1V is dependable and efficient; on a 129×129 grid divergence occurs for certain convection directions, due to bad coarse grid approximations. The ILLU smoothing process comes so close to being exact for small ε, using upwind differences, that notwithstanding the bad coarse grid corrections convergence is very fast on grids of all sizes.

The coarse grid approximations are improved by the P9' or P9" prolongations and restriction, resulting in fast convergence on grids of all sizes.

For this class of problems the ILLU smoothing process is more efficient than ILU and AZ. The sawtooth cycle is more efficient than the V-cycle in the cases considered.

10. RESULTS FOR THE ANISOTROPIC DIFFUSION AND MIXED DERIVATIVE TEST PROBLEMS.

For test problems (3.2), (3.3) and (3.4) the exact numerical solution is chosen as $u^{\ell} = x_1(1-x_1)x_2(1-x_2).10^6$. The boundary conditions are of Dirichlet type. They are not eliminated but treated as in the preceding section with p=40. The initial guess is identically zero. The Euclidean norm of the residue is measured, and the average reduction factor over iterations 2-10 called ρ is reported. Two smoothing processes are tested, namely ILU and HZ.

The cost of ILU is about 1.5 that of HZ (cf. [14] table 9.6b). The following results are obtained for test problem (3.2).

ε	ρ	ρ	ε	ρ	ρ
10^6	0.000	0.241	10^{-1}	0.088	0.723
10^4	0.061	0.141	10^{-2}	0.071	0.766
10^3	0.345	0.116	10^{-3}	0.040	0.747
10^2	0.495	0.096	10^{-4}	0.000	0.746
10	0.241	0.152	10^{-6}	0.000	0.745
1	0.072	0.379			
	ILU	HZ		ILU	HZ

Table 10.1
Average reduction factors for test problem (3.2).

The finest grid has dimension 65×65, the coarsest 3×3. The ILU results have been obtained with the MGD1V FORTRAN code; for HZ the same multigrid strategy is used.

For ε small we have strong coupling in the x_2-direction, and we should use vertical lines for ZEBRA, obtaining the same results as for $1/\varepsilon$. If ε varies widely in the region, taking on both large and small values, one

should use AZ, if one prefers the ZEBRA smoothing process.

Based on the smoothing analysis results of [10], appendix B, ILU should not work for large ε, because the smoothing factor tends to 1 as $\varepsilon \to \infty$. The local Fourier two-grid analysis results of [14], table 9.6a, lead to the same conclusion. However, table 10.1 shows that ILU works well for this case. The resolution of this apparent paradox is to be sought in the role of the boundary conditions. These are neglected in local Fourier analysis, which is carried out on an infinite region. However, for $\varepsilon << 1$ and $\varepsilon >> 1$ the influence of the boundary conditions extends far into the region. Apparently, the Fourier modes which are not treated well by ILU are excluded by the boundary conditions. The possible influence of boundary conditions should be kept in mind when interpreting Fourier analysis results. It is desirable that predictions about the performance of a multigrid method are corroborated by results obtained by actual use of the method. Such results are necessarily of a statistical nature; the average reduction factors obtained depend on the right-hand-side, the initial guess and the number of iterations.

As a further illustration of the effectiveness of ILU and ZEBRA for this test problem, we present the following one-grid results.

ε	ρ	ρ	ε	ρ	ρ
10^6	0.000	0.127	10^{-1}	0.912	0.993
10^4	0.071	0.075	10^{-2}	0.526	0.993
10^3	0.424	0.205	10^{-3}	0.057	0.992
10^2	0.858	0.793	10^{-4}	0.000	0.992
10	0.966	0.969	10^{-6}	0.000	0.992
1	0.976	0.992			
	ILU	HZ		ILU	HZ

Table 10.2 Average reduction factors on a single 65×65 grid for test problem (3.2).

For $\varepsilon << 1$ and $\varepsilon >> 1$ ILU is almost exact; for $\varepsilon \approx 1$ we need coarse grid corrections to accelerate convergence.

Next, we turn to test problems (3.3) and (3.4). The following results are obtained.

Test problem (3.3)	ρ	ρ	Test problem (3.4)	ρ	ρ
	0.074	0.663		0.174	0.400
	ILU	HZ		ILU	HZ

Table 10.3 Average reduction factors for test problems (3.3) and (3.4).

again, the finest grid has dimension 65×65, the coarsest 3×3; the same multigrid strategy has been used as for table 10.1.

As a further illustration of the effectiveness of ILU and HZ we present the following one-grid results for these test problems.

Test problem (3.3)	ρ 0.729 ILU	ρ 0.980 HZ	Test problem (3.4)	ρ 0.954 ILU	ρ 0.979 HZ

Table 10.4 Average reduction factors on a single 65×65 grid for test problems (3.3) and (3.4).

From these experiments we draw the following conclusions.

For test problem (3.2), the ZEBRA smoothing process is efficient with the lines chosen vertically if $\varepsilon \ll 1$ and horizontally if $\varepsilon \gg 1$. The ILU smoothing process is efficient regardless of the value of ε. If for ILU one chooses the x_2-axis in the direction of strong coupling with the grid point ordering given in section 3, or if one orders the grid points and unknowns along vertical lines for $\varepsilon \gg 1$ or along horizontal lines (as shown in section 3) for $\varepsilon \ll 1$, ILU is considerably more efficient than ZEBRA. We have no results for problem (3.2) in which ε is variable and takes on both large and small values, but expect that both ILU and AZ will work well.

For test problems (3.3) and (3.4) ILU is more efficient than HZ.

As yet, we have no results for ILLU for test problems (3.2)-(3.4).

Comparing tables 10.1 and 10.2, and tables 10.3 and 10.4 we see that the coarse grid corrections are effective for these test problems. Therefore we expect the results that have been presented to be representative of the convergence behaviour on still finer grids, and that the sort of trouble encountered in this respect in section 9 does not occur for the test problem treated here.

More results for these test problems with the MGD1 method have been published in [18, 19].

11. FINAL REMARKS

The development of multigrid fast solvers has been shown to be feasible. It is desirable that when using such a solver the user has only to specify the problem on the computational grid that he wants to use, and that he does not need to involve himself with the underlying algorithm. The MGD1 method described satisfies these requirements. A multigrid fast solver based on a similar design philosophy has been described in [5], where a "black box" FORTRAN code called BOXMG is presented, that works for discontinuous coefficients and arbitrary regions. Also in BOXMG the user is asked to control the multigrid algorithm to a certain extent, and given a choice of smoothing processes of point and line Gauss-Seidel type. Other available software is the collection of multigrid solution modules MG00 and MG01 [22], [23].

The numerical experiments that have been described make it plausible, that the method convergences fast for the general elliptic equation discretized on a square, with continuous coefficients, including non-self-adjoint problems, a mixed derivative and the limiting cases of strong convection and strongly anisotropic diffusion. The method has been implemented in the portable FORTRAN code MGD1V. This code auto-vectorizes satisfactorily. A special CYBER-205 version called MGD1D is identical to MGD1V except for a few statements, which have been replaced by calls to the CYBER STACKLIB library.

Acknowlegement We are indebted to Mr. W. Lioen, who performed some of the computations reported in this paper.

Note The MGD1V and MGD1D codes can be obtained by sending a tape to the third author.

References

1 R.E. Alcouffe, A. Brandt, J.E. Dendy, Jr., J.W. Painter, The multi-grid method for the diffusion equation with strongly discontinuous coefficients. SIAM J. Sci. Stat. Comp. 2, 430-454, 1981.

2 A. Brandt, Multi-level adaptive solutions to boundary-value problems. Math. Comp. 31, 333-390, 1977.

3 A. Brandt, Multi-level adaptive solutions to singular perturbation problems. In: P.W. Hemker and J.J. Miller (eds.), Numerical analysis of singular perturbation problems, pp. 53-142. Academic Press, New York, 1979.

4 A. Brandt, Guide to multigrid development. In: W. Hackbusch and U. Trottenberg (eds.), Multigrid methods. Proceedings, Koln-Porz, 1981. Lect. Notes in Math. 960, pp. 280-312. Springer-Verlag, Berlin etc. 1982.

5 J.E. Dendy, Jr., Black box multigrid. J. Comp. Phys. 48, 366-386, 1982.

6 W. Hackbusch, On the multi-grid method applied to difference equations. Computing 20, 291-306, 1978.

7 P.W. Hemker, The incomplete LU-decomposition as a relaxation method in multigrid algorithms. In: J.J.H. Miller (ed.), Boundary and interior layers - computational and asymptotic methods, pp. 306-311. Boole Press, Dublin, 1980.

8 P.W. Hemker, On the comparison of line-Gauss-Seidel and ILU relaxation in multigrid algorithms. In: J.J.H. Miller (ed.), Computational and asymptotic methods for boundary and interior layers, pp. 269-277. Boole Press, Dublin, 1982.

9 R. Kettler, A study of the applicability of the multiple grid method in reservoir simulation. Part II. Master's thesis, Delft University of Technology, Nov. 1980.

10 R. Kettler, Analysis and comparison of relaxation schemes in robust multigrid and preconditioned conjugate gradient methods. In: W. Hackbusch and U. Trottenberg (eds.), Multigrid methods. Proceedings, Köln-Porz, 1981. Lect. Notes in Math. 960, pp. 502-534, Springer-Verlag, Berlin etc., 1982.

11 R. Kettler, J.A. Meijerink, A multigrid method and a combined multigrid-conjugate gradient method for elliptic problems with strongly discontinuous coefficients in general domains. KSEPL Publication 604, Kon. Shell Expl. and Prod. Lab., Rijswijk, The Netherlands, 1981.

12 J.A. Meijerink, H.A. van der Vorst, An iterative solution method for linear systems of which the coefficient matrix is a symmetric M-matrix. Math. Comp. 31, 148-162, 1977.

13 R.A. Nicolaides, On some theoretical and practical aspects of multigrid methods. Math. Comp. 33, 933-952, 1979.

14 K. Stüben, U. Trottenberg, Multigrid methods: fundamental algorithms, model problem analysis and applications. In : W. Hackbusch and U. Trottenberg (eds.), Multigrid methods. Proceedings, Köln-Porz, 1981. Lect. Notes in Math. 960, pp. 1-176, Springer-Verlag, Berlin etc., 1982.

15 R. Temam, Numerical analysis. D. Reidel Publ. Comp., Dordrecht/Boston, 1973.

16 P. Wesseling, P. Sonneveld, Numerical experiments with a multiple grid and a preconditioned Lanczos type method. In R. Rautmann (ed.): Approximation methods for Navier-Stokes problems. Proceedings, Paderborn 1979. Lect. Notes in Math. 771, pp. 543-562, Springer-Verlag, Berlin etc., 1980.

17 P. Wesseling, The rate of convergence of a multiple grid method. In:

G.A. Watson (ed.), Numerical analysis. Proceedings, Dundee, 1979. Lect. Notes in Math. 773, pp. 164-184, Springer-Verlag, Berlin etc., 1980.

18 P. Wesseling, Theoretical and practical aspects of a multigrid method. SIAM J. Sci. Stat. Comp. 3, 387-407, 1982.

19 P. Wesseling, A robust and efficient multigrid method. In : W. Hackbusch and U. Trottenberg (eds.), Multigrid methods. Proceedings, Köln-Porz, 1981. Lect. Notes in Math. 960, pp. 614-630, Springer-Verlag, Berlin etc., 1982.

20 P.M. de Zeeuw, E.J. van Asselt, The convergence rate of multi-level algorithms applied to the convection-diffusion equations. Report NW 142/82, Mathematical Center, Amsterdam, 1982.

21 E.J. van Asselt, The multigrid method and artificial viscosity. In: W. Hackbusch and U. Trottenberg (eds.), Multigrid methods. Proceedings, Köln-Porz, 1981. Lect. Notes in Math. 960, pp. 313-326, Springer-Verlag, Berlin etc. 1982.

22 H. Foerster, K. Witsch, Multigrid software for the solution of elliptic problems on rectangular domains: MGOO (Release 1). In: W. Hackbusch and U. Trottenberg (eds.), Multigrid methods. Proceedings, Köln-Porz, 1981. Lect. Notes in Math. 960, pp. 427-461, Springer-Verlag, Berlin etc., 1982.

23 K. Stüben, U. Trottenberg, K. Witsch, Software development based on multigrid techniques (to appear).