MULTIGRID AND CONJUGATE GRADIENT ACCELERATION OF BASIC ITERATIVE METHODS

P. Sonneveld and P. Wesseling
*(Department of Mathematics and Informatics
Delft University of Technology, The Netherlands)*

and

P.M. de Zeeuw
*(Centre for Mathematics and Computer Science
Amsterdam, The Netherlands)*

1. INTRODUCTION

Let a discretisation of a partial differential equation be denoted by

$$Ay = b \tag{1.1}$$

This is a system of linear equations.

Stationary single-step iterative methods for solving (1.1) can be written as

$$y^{n+1} = y^{n} + B(b - Ay^{n}) \tag{1.2}$$

where the matrix B, which often is not formed explicitly, depends on the iterative method that is chosen.

If the discretisation is nonlinear, it is assumed that some linearisation leads to a linear system (1.1). The iterations (1.2) will then be inner iterations within an outer iteration on the nonlinearity.

In practical applications the number of unknowns in (1.1) can be quite large, and simple and straightforward iterative methods normally converge slowly. A significant recent development in numerical mathematics is the rise of preconditioned conjugate gradient methods (CG) and multigrid methods (MG). With respect to linear problems, both can be regarded as acceleration techniques for basic iterative methods. This provides us with a unifying point of view.

The purpose of this paper is to briefly review CG and MG from the viewpoint just mentioned, and to introduce the reader to the literature.

For completeness it should be mentioned that MG can also be applied directly to nonlinear problems, and this may lead to a more efficient solution method than linearisation followed by application of a fast linear solver. See (Hackbusch and Trottenberg, 1982) for a general introduction to MG. Up-to-date accounts of CG are given in (Hageman and Young, 1981; Golub and van Loan, 1983).

2. BASIC ITERATIVE METHODS

"Basic iterative method" (BIM for short) is a loose appellation for methods of type (1.2) that are simple and easy to implement. For a review of BIMs (including some that are not so simple) see (Young, 1971, or Varga, 1962). Examples are the Gauss-Seidel (GS) and SOR (successive over-relaxation) methods. The computational complexity of these methods for a typical problem such as the two-dimensional Poisson equation is $O(N^2)$ and $O(N^{3/2})$, respectively, with N the number of grid points. These numbers explain both the popularity of SOR in the sixties, and the large computer times needed for large N. Accelerated by CG or MG, the computational complexity becomes $O(N^{5/4})$ (conjecture) or $O(N)$, respectively, as will be discussed later.

The most popular BIMs used with CG and MG are GS methods and incomplete factorisation (IF) methods.

The reader is assumed to be familiar with GS methods. They come in various orderings (lexicographic, red-black etc.) and may be pointwise or blockwise (by lines or planes mostly). We give a short outline of IF methods.

An incomple LU-factorization (ILU) of the matrix A consists of a lower triangular matrix L and an upper triangular matrix U such that

$$LU = A + C \qquad (2.1)$$

where C represents the error matrix. With C = O (complete factorization) L and U usually are much less sparse than A, which leads to large storage and computer time requirements. By allowing $C \neq O$ sparsity of L and U can be obtained. For example, if A represents a typical two-dimensional 7-point finite difference stencil of a second order elliptic partial

difference operator on a rectangular grid, the sparsity patterns of L, U, A and C could be as in Fig. 2.1.

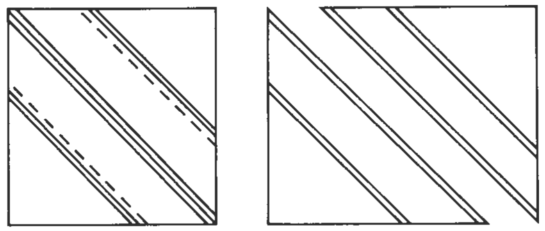

Fig. 2.1. Sparsity patterns. - - - : C

For simple formulae for the computation of L, U and C on a rectangular computational grid and theoretical results on the existence of L and U see (Meijerink and van der Vorst, 1977, 1981, Wesseling, 1982A, Sonneveld et al., 1985).

Equation (2.1) leads to the following iterative method for solving (1.1):

$$y^{n+1} = y^n + (LU)^{-1}(b - Ay^n) \qquad (2.2)$$

Using (2.1) this can be rewritten as

$$y^{n+1} = (LU)^{-1}(b + Cy^n) \qquad (2.3)$$

Hence, A is not needed, and L and U can be stored at the location of A. Usually C is not stored but computed. Hence, no extra storage is required. Because C is very sparse (cf. Fig. 2.1) the right hand side of (2.3) comes cheap. One iteration with (2.3) takes 18N operations, generation of L and U takes 21N operations, with N the number of unknowns: see for example (Wesseling, 1982A).

Incomplete block LU-factorization (IBLU or ILLU, incomplete line LU-factorization) of A goes as follows. On a rectangular grid with n vertical and n horizontal lines the matrix A has the following structure:

$$A = \begin{bmatrix} B_1 & U_1 & & & \\ L_2 & B_2 & U_2 & & \\ & L_3 & B_3 & U_3 & \\ & & \cdot & \cdot & \cdot \\ & & \cdot & \cdot & \cdot \\ & & & L_n & B_n \end{bmatrix} \qquad (2.4)$$

with L_i, B_i and U_i $m \times m$ matrices; B_i is tridiagonal; L_i and U_i are lower and upper bidiagonal, with sparsity patterns $\{(j,j-1), (j,j)\}$ and $\{(j,j), (j,j+1)\}$ respectively. There exists a matrix D such that

$$A = (L + D)D^{-1}(D + U) \qquad (2.5)$$

where

$$L = \begin{bmatrix} O & & & & \\ L_2 & O & & & \\ & L_3 & O & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & L_n & O \end{bmatrix}, \quad U = \begin{bmatrix} O & U_1 & & & \\ & O & U_2 & & \\ & & \cdot & \cdot & \\ & & & \cdot & \cdot \\ & & & O & U_{n-1} \\ & & & & O \end{bmatrix},$$

$$D = \begin{bmatrix} D_1 & & & & \\ & D_2 & & & \\ & & \cdot & & \\ & & & \cdot & \\ & & & & D_n \end{bmatrix}$$

Equation (2.5) is sometimes called a line LU-factorization because the blocks in L, D and U correspond to (in this case horizontal) lines in the grid. Equation (2.5) can be rewritten as

$$A = L + D + U + LD^{-1}U \qquad (2.6)$$

One finds that $LD^{-1}U$ is the following block-diagonal matrix:

$$LD^{-1}U = \begin{bmatrix} O & & & \\ & L_2D_1^{-1}U_1 & & \\ & & \cdot & \\ & & & \cdot \\ & & & & L_nD_{n-1}^{-1}U_{n-1} \end{bmatrix} \qquad (2.7)$$

From (2.6) and (2.7) we deduce the following algorithm for the computation of D:

$$D_1 = B_1, \; D_i = B_i - L_iD_{i-1}^{-1}U_{i-1}, \; i = 2,3,\ldots n \qquad (2.8)$$

The matrix D_i^{-1} is a full $m \times m$ matrix, which causes the cost of a line LU-factorization to be $O(nm^3)$, as for standard LU-factorization. An IBLU factorization is obtained if we replace $L_iD_{i-1}^{-1}U_{i-1}$ by its tridiagonal part. Thus, algorithm (2.8) is replaced by:

$$\tilde{D}_1 = B_1, \; \tilde{D}_i = B_i - \text{tridiag}\,(L_i\tilde{D}_{i-1}^{-1}U_{i-1}), \; i = 2,3,\ldots,n. \qquad (2.9)$$

The IBLU factorization of A is now defined to be

$$A = (L + \tilde{D})\tilde{D}^{-1}\,(\tilde{D} + U) + R \qquad (2.10)$$

with R the error matrix, and $\tilde{D}$ the block diagonal matrix with blocks $\tilde{D}_i$.

Detailed formulae for the computation of $\tilde{D}$ are given by (Concus et al., 1985; Sonneveld et al., 1985; Axelsson et al., 1983; Meijerink, 1983; Underwood, 1976).

From (2.10) the following iterative method is obtained:

$$(L + \tilde{D})\tilde{D}^{-1}(\tilde{D} + U)\delta y = b - Ay^n, \quad y^{n+1} = y^n + \delta y^n \qquad (2.11)$$

The cost of one iteration with (2.11) is 37N operations; the cost of $\tilde{D}$ and its triangular factorization is 36N operations. Storage of 6N reals is required for the triangular factorization of $\tilde{D}$ and for $\tilde{D}$. For details see (Sonneveld ed al., 1985).

Both ILU and IBLU can be implemented efficiently on vector computers: see (van der Vorst, 1982, 1985; Meurant, 1984; Hemker et al., 1983). Variants of ILU are discussed by Behie and Forsyth (1984), Gustafsson (1978) and Manteuffel (1979).

Extension of ILU to three dimensions is straightforward, and has been tested in combination with CG and MG by Behie and Forsyth (1983). A three-dimensional version of IBLU has been proposed by Meijerink (1983) and Kettler and Wesseling (1985).

3. CONJUGATE GRADIENT ACCELERATION OF BASIC ITERATIVE METHODS

For an introduction to conjugate gradients (CG) (and Tchebychev) methods, see (Hageman and Young, 1981, or Golub and van Loan, 1983). Here only a brief outline is given.

One way to look at CG is as follows. Assuming the matrix A to be large and sparse we allow ourselves to use A only for multiplication with vectors and nothing else. This means that polynomials in A can be built. A rather general algorithm then is

$$y^{n+1} = y^n + \alpha_n p^n, \qquad p^n = \theta_n(A) r^0 \qquad (3.1)$$

Here α_n is a scalar to be determined, θ_n is a polynomial of degree n, and $r^0 = b - Ay^0$ is the initial residue. We find

$$r^{n+1} = b - Ay^{n+1} = \phi_{n+1}(A) r^0 \qquad (3.2)$$

with

$$\phi_{n+1}(A) = I - A\{\alpha_n\theta_n(A) + \alpha_{n-1}\theta_{n-1}(A) + \dots + \alpha_n\theta_0(A)\}r^0 \qquad (3.3)$$

Hence

$$\phi_n \in \Pi_n^1, \qquad (3.4)$$

$$\Pi_n^1 = \{\psi_n | \psi_n(0) = 1, \ \psi_n \text{ is polynomial of degree} \leq n\} \qquad (3.5)$$

The distinguishing property of CG is that ϕ_n is constructed such that

$$||\phi_n(A) r^0|| \leq ||\psi_n(A) r^0||, \quad \forall \ \psi_n \in \Pi_n^1 \qquad (3.6)$$

Depending on the choice of norm, different CG variants are obtained.

By itself CG does not achieve an impressive rate of convergence. It comes into its own when it is combined with preconditioning. Let B be the preconditioning matrix. Then a preconditioned CG method is given by:

$$\begin{aligned} &p^{-1} = 0, \ r^0 = b - Ay^0, \\ &p^n = Br^n + \beta_n p^{n-1}, \ \beta_n = \rho_n/\rho_{n-1}, \ \rho_n = r^{n^T} Br^n, \\ &y^{n+1} = y^n + \alpha_n p^n, \ \alpha_n = \rho_n/\sigma_n, \ \sigma_n = p^{n^T} Ap^n, \\ &r^{n+1} = r^n - \alpha_n Ap^n \end{aligned} \qquad (3.7)$$

see (Hestenes, 1956). With B = I one obtains an unpreconditioned CG method. In this case the norm in (3.6) has been chosen as follows:

$$||r||^2 = r^T A^{-1} r \qquad (3.8)$$

Now we will explain why (3.7) can be regarded as an acceleration of the BIM (1.2), under the assumption that B is SPD, so that one may write $B = EE^T$. By substitution of $r = E^{-T}\tilde{r}$, $p = E\tilde{p}$, $y = E\tilde{y}$ one finds that (3.7) is CG applied to the preconditioned system $\tilde{A}\tilde{y} = \tilde{b}$ with $\tilde{b} = E^T b$, $\tilde{A} = E^T AE$. According to (3.2), (3.6) we have

$$E^T r^n = \phi_n(E^T AE) E^T r^0 \qquad (3.9)$$

with

$$||\phi_n(E^T AE) E^T r^0|| \leq ||\psi_n(E^T AE) E^T r^0||, \quad \forall \psi_n \in \Pi_n^1 \qquad (3.10)$$

From (1.2) it follows that

$$E^T r^n = E^T \psi_n(AEE^T) E^{-T} E^T r_0, \psi_n(x) = (1 - x)^n \in \Pi_n^1 \qquad (3.11)$$

Since $E^T(AEE^T)^k E^{-T} = (E^T AE)^k$, $\forall$ k eq. (3.11) can be rewritten as

$$E^T r^n = \psi_n(E^T AE) E^T r^0 \qquad (3.12)$$

Comparison of (3.9) - (3.12) shows that (3.7) will converge at least as fast as (1.2). The computation of Br^n in (3.7) is performed by means of one iteration with (1.2), since

$$Br^n = y^{n+1} - y^n \tag{3.13}$$

The rate of convergence of CG has been studied in a number of publications. From (3.6) one may deduce, as in (Daniel, 1967), that the required number of iterations n to reduce the residue by a factor ε satisfies

$$n \geq \frac{1}{2} \ln \frac{\varepsilon}{2} \operatorname{cond}_2(BA)^{\frac{1}{2}} \tag{3.14}$$

with cond_2 the condition number in the Euclidean norm. For a detailed study of the rate of convergence see (van der Sluis and van der Vorst, 1985).

An effective and popular preconditioning is provided by incomplete Cholesky factorization, leading to the ICCG method, proposed by Meijerink and van der Vorst (1977). Its effectiveness is explained by taking a look at the spectrum of BA. Gustafsson (1978) proves that with a modified form of incomplete Cholesky factorization for the Poisson equation in two dimensions

$$\operatorname{cond}_2(BA) = O(1/h) \tag{3.15}$$

so that because of (3.14) the computational cost of $O(N^{5/4})$, with N the number of gridpoints. Compared with older methods, very impressive rates of convergence are obtained with ICCG for a number of difficult problems by Kershaw (1978). An implementation leading to greater efficiency is given by Eisenstat (1981).

The restriction of CG to SPD problems is of course a severe one. Various CG type methods have been proposed for non-SPD systems. An example is the CGS (conjugate gradients squared) method, proposed by Sonneveld (1984). This method is defined as follows, in its preconditioned form:

$$\begin{aligned}
f^0 &= b - Ay^0 , \quad g^{-1} = h^0 = 0, \\
u &= Bf^n + \beta_n h^n , \quad g^n = u + \beta_n(\beta_n g^{n-1} + h^n), \\
h^{n+1} &= u - \alpha_n BAg^n , \quad y^{n+1} = y^n + \alpha_n(u + h^{n+1}) , \\
f^{n+1} &= f^n - A\alpha_n (u + h^{n+1}) ,
\end{aligned} \tag{3.16}$$

with

$$\alpha_n = \rho_n/\sigma_n , \quad \beta_0 = 0 , \quad \beta_n = \rho_n/\rho_{n-1} ,$$

$$\sigma_n = \tilde{r}^{0^T} Bf^n , \quad \sigma_n = \tilde{r}^{0^T} BAg^n$$

where $\tilde{r}^0$ is a vector to be chosen.
For a justification of this method see (Sonneveld, 1984; Sonneveld et al., 1985). It is related to the biconjugate gradient (bi-CG) method of Fletcher (1976). With bi-CG one has

$$r^n = \phi_n(BA)r^0 \tag{3.17}$$

With CGS one obtains at almost no extra cost

$$r^n = \phi_n^2(BA)r^0 \tag{3.18}$$

The polynomials in (3.17) and (3.18) are the same. It follows that if bi-CG converges, then CGS converges faster (the S in "CGS" is inspired by the squaring of ϕ_n). Furthermore, unlike bi-CG, CGS does not use A^T.

Bi-CG and CGS are but two examples of extensions of CG to non-SPD systems. We will not review other extensions that have been proposed, but restrict ourselves to mentioning the publications of Concus and Golub (1976), Vinsome (1976), Widlund (1978), Axelsson (1980), van der Vorst (1981) and Young and Jea (1980). A somewhat related class of methods is formed by the Tchebychev methods: see (Manteuffel, 1977, 1978; Hageman and Young, 1981). These methods can be very efficient, but their effectiveness depends on the choice of certain parameters.

Unlike the SPD case, in the non-SPD case the theory is far from complete. For CGS applied to a general system a rule of thumb is, that with ILU or IBLU preconditioning good convergence may be expected if A satisfies

$$a_{ii} \geq -\sum_{j \neq i} a_{ij} \ , \ a_{ij} \leq 0 \ , \ j \neq i \tag{3.19}$$

Sonneveld (1984) and Sonneveld et al. (1985) estimate the cost of a CGS iteration to be 60 flops (ILU preconditioning) or 88 flops (IBLU preconditioning) per grid point.

4. MULTIGRID ACCELERATION OF BASIC ITERATIVE METHODS

The basic ideas of MG are quite general and have wide range of application, including other fields than partial differential equations. MG can be used not only to accelerate iterative methods, but also, for example, to formulate novel ways to solve nonlinear problems, including bifurcation problems and eigenvalue problems, or to devise algorithms that construct adaptive discretizations. The volume edited by Hackbusch and Trottenberg (1982) presents a useful survey of all aspects of MG. See also (Brandt, 1984).

We restrict ourselves here to the one aspect of MG mentioned in the title of this paper. This makes it possible to simplify MG, and to distinguish situations where its effectiveness is guaranteed. The significance of MG as an acceleration technique springs from the fact, that in principle a computational complexity of O(N) can be achieved. This has been proved rigorously under quite general circumstances; see (Hackbusch, 1982) for a survey of MG convergence theory. Work in this area is still going on.

If the BIM (1.2) converges, it usually has the property, exploited by MG, that the non-smooth part of error and residue is annihilated rapidly, whereas it takes many iterations (more and more as the mesh size decreases) to get rid of the smooth part. The property of smoothness will be defined precisely shortly. The fundamental MG idea is to approximate the smooth part of the error on coarser grids. In the MG context the BIM (1.2) is called a smoothing process.

A two-grid method can be defined and explained as follows. Let G be a fine grid and $\bar{G} \subset G$ a coarse grid. Let the sets of grid functions $G \to \mathbb{R}$ and $\bar{G} \to \mathbb{R}$ be denoted by Y and $\bar{Y}$, respectively. Let the coarse grid approximation of (1.1) be given by

$$\bar{A}\bar{y} = \bar{b} \tag{4.1}$$

Furthermore, let there be given a prolongation operator P and a restriction operator R:

$$P : \bar{Y} \to Y \ , \ R : Y \to \bar{Y} \tag{4.2}$$

A two-grid method for the acceleration of the BIM (1.2) can be formulated as follows. Let y^j be the current iterand, and let $y^{j+\frac{1}{2}}$ be the result of applying a coarse grid correction to y^j:

$$y^{j+\frac{1}{2}} = y^j + P\bar{A}^{-1}R(b - Ay^j) \tag{4.3}$$

where we assume for the time being that the coarse grid problem is solved exactly. For the residue $r^j = b - Ay^j$ we find:

$$r^{j+\frac{1}{2}} = (I - AP\bar{A}^{-1}R)r^j \tag{4.4}$$

We now make the following choice for $\bar{A}$, called Galerkin approximation:

$$\bar{A} = RAP \tag{4.5}$$

A more obvious choice of $\bar{A}$ would be to discretise the differential equation on the coarse grid $\bar{G}$, but (4.5) leads to a more elegant exposition of MG principles, and has other advantages as well, to be mentioned later. It now follows from (4.4) that

$$r^{j+\frac{1}{2}} \in \mathrm{Ker}(R) \tag{4.6}$$

as noted by Hemker (1982) and McCormick (1982). In other words, $r^{j+\frac{1}{2}} \perp \mathrm{Ker}^{\perp}(R)$, which justifies the appellation "Galerkin approximation" for (4.5). Following Hemker (1982) we relate the concept of smoothness to Ker(R):

Definition 4.1 The set of R-smooth grid functions is $\mathit{Ker}^{\perp}(R)$.

This definition makes sense intuitively, since in practice R consists of taking weighted averages with positive weights over the fine grid, so that a grid function in Ker (R) has many sign changes, and hence may be called non-smooth.

It remains to annihilate the non-smooth part of r, and this is done in the second part of two-grid iteration, called smoothing. This is done with the BIM (1.2):

$$y^{j+1} = y^{j+\frac{1}{2}} + B(b - Ay^{j+\frac{1}{2}}) \tag{4.7}$$

and we find:

$$r^{j+1} = (I - AB)r^{j+\frac{1}{2}} \tag{4.8}$$

The projection operator on Ker(R) is $I - R^T(RR^T)^{-1}R$, and we conclude from (4.8) and (4.6) that

$$r^{j+1} = (I - AB)(I - R^T(RR^T)^{-1}R)r^{j+\frac{1}{2}} \tag{4.9}$$

This leads us to the following definition:

Definition 4.2 The R-smoothing factor of the smoothing process (1.2) is

$$\rho_R = ||(I - AB)(I - R^T(RR^T)^{-1}R||$$

The concepts of smoothness and smoothing factor can also be related to the range of P, or be introduced by means of Fourier analysis as proposed by Brandt (1977). For definitions, see (Sonneveld et al., 1985). Fourier analysis leads to less precise exposition, but to smoothing factors that are easy to compute. See (Brandt, 1982, and Kettler, 1982) for more details.

If we postulate that

$$||I - AP\bar{A}^{-1}R|| \leq 1 \tag{4.10}$$

(which holds approximately if P and R are accurate enough) then we may conclude from (4.4) and (4.9) that

$$||r^{j+1}|| \leq ||(I - AB)(I - R^T(RR^T)^{-1}R)||\,||r^j|| \tag{4.11}$$

Without coarse grid correction, the residue reduction factor would be $||I - AB||$. Equation (4.11) shows how MG accelerates the convergence of the BIM (1.2), if the BIM reduced non-smooth residue components efficiently.

Choosing $\bar{A}$ according to (4.5) has the advantage that the user needs to specify the matrix and the right hand side on the finest grid only. Hence a multigrid code can be programmed such that it is perceived by the user just like any other subroutine for solving linear algebraic systems. Furthermore, (4.5) provides automatically a sound "homogenization" in cases where the coefficients or the right hand side vary rapidly. With difference approximation the implementation of the boundary conditions on the coarse grids is quite often not trivial. On the other hand, depending on the problem, computing $\bar{A}$ with (4.5) may be more expensive than constructing a finite difference approximation. Also $\bar{A}$ has to be stored (perhaps to be overwritten at a later stage by its incomplete factorization). With adaptive discretisation (4.5) is not feasible, since one does not know a priori what the finest grid will be. Results of numerical experiments comparing coarse grid Galerkin and finite difference approximation are given by Wesseling (1982A).

For second order equations prolongation may be defined by linear interpolation. If the "full multigrid" schedule is used, second order interpolation is also needed at certain stages in the schedule: see (Brandt, 1977). For restriction one may safely take $R = P^T$. For discussions of various other possibilities, and comparative experiments, see for example (Brandt, 1977, 1982; Stuben and Trottenberg, 1982; Wesseling, 1982A). However, when the coefficients in the given differential equation are strongly discontinuous one should use matrix-dependent prolongation instead of linear interpolation, in order to take into account the fact, that the solution is locally linear only in a piece-wise fashion. One may still use $R = P^T$. See (Alcouffe et al., 1981; Kettler, 1982).

The two grid method described above becomes a multigrid method if exact solution of the coarse grid is replaced by approximate solution, using MG with coarser grids. In the MGD-family of MG codes we do this with one two-grid iteration employing an additional coarser grid with doubled mesh-size, and so on recursively, until the coarsest grid (usually a 3 x 3 grid) is reached, where one iteration is performed according to (1.2). The resulting MG method is said to be of sawtooth type, because its schedule is represented in a natural way by the schematic of Fig. 4.1, which is a sawtooth curve.

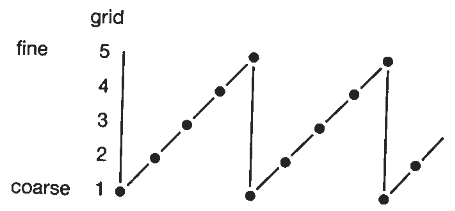

Fig. 4.1. Sawtooth multigrid schedule.
A dot represents a smoothing step.

Various more general MG schedules have been described, see for example (Brandt, 1977, 1982), (Stüben and Trottenberg, 1982). Some comparative experiments are described by Wesseling (1982A). The sawtooth schedule is the simplest possible MG schedule. One may wonder whether such a simple schedule can handle a sufficiently large variety of cases. Experience indicates that the answer is affirmative, see for example the experiments carried out by Wesseling and Sonneveld (1980), Kettler (1982), Wesseling (1982A,B), Hemker et al. (1983), McCarthy (1983), Sonneveld et al. (1985), and the application to transonic potential flow by Nowak and Wesseling (1984). We think that with an effective smoothing process and accurate coarse grid approximation, a simple MG schedule suffices for linear problems.

The MGD-family of MG codes consists at the moment of the subroutines MGD1 and MGD5. These are meant for solving finite difference discretisations of a general second order partial differential operator (possibly not self-adjoint, with a mixed derivative allowed) on a two dimensional rectangular uniform grid. The user provides only the matrix and the right hand side on the finest grid. Because the user is not allowed to influence these subroutines by setting parameters etc, they are called autonomous. MGD1 and MGD5 work like a "black box". MGD1 uses ILU smoothing, MGD5 employs IBLU. As a result, MGD5 is more robust. These programs are portable. In the near future, MGD1 will be available in the NAG library. Versions MGD1V and MGD5V have been designed for auto-vectorization on CRAY and CYBER-205, without sacrificing much on scalar machines. More details, and CPU-time measurements on CYBER-170, CYBER-205 and CRAY-1, and some design considerations related to these MG subroutines may be found in (Wesseling, 1982B, Hemker et al., 1983, 1984, 1985, Sonneveld et al., 1985). The MGD software may be obtained by sending a magnetic tape to the second author of the present paper.

MG software (the MGOO package) is also made available by the group at GMD, Bonn, see (Foerster and Witsch, 1982; Stüben et al., 1984). MGOO is for self-adjoint second order elliptic partial differential equations on a two dimensional rectangular domain. Unlike the MGD codes, the user is allowed some freedom in choosing the MG algorithm. There is a choice in MG schedules, smoothing processes and restriction operators. For the Poisson equation an especially fast version is available. A mixed derivative is not allowed.

Dendy (1982) describes a "black box multigrid" method that can handle self-adjoint equations with strongly discontinuous coefficients; the software mentioned before can handle this only when the discontinuities occur on grid lines that belong to at least a few coarse grids. The user has a choice of multigrid schedule and smoothing processes.

It is to be expected that results of comparative numerical experiments with the software mentioned above will become available in the near future.

5. DISCUSSION AND FINAL REMARKS

It will be clear from the foregoing that CG has a sound theoretical basis only in the SPD case. For MG the O(N) complexity is theoretically well-founded only if one regards the coefficients in the differential equation as fixed. However, in practice one often has a need for numerical methods the properties of which remain invariant as a coefficient tends to a limit. Examples are the convection-diffusion equation at high Péclet number:

$$u_i\phi_{,i} - \varepsilon\phi_{,ii} = f \tag{5.1}$$

with $1/\varepsilon$ the Péclet number, or the Navier-Stokes equations at high Reynolds number, or the anisotropic diffusion equation:

$$-\varepsilon\phi_{,11} - \phi_{,22} = f \tag{5.2}$$

Equation (5.2) also models the effect of having a computational grid with cells of very high aspect ratio. Testing in practice seems to be the only way to determine how a method works in cases like this. In fact, also in easier cases where $\varepsilon = O(1)$ practical tests are needed to find out what the efficiency of a method is; there may be large differences in computer time between two methods that both have complexity O(N).

Therefore there is a need for test problems of sufficient generality such that one may have a reasonable degree of confidence in those methods that have done well on these test problems. One important test equation is (5.1), with u_i in "all" possible directions, for example $u_1 = \cos\alpha$, $u_2 = \sin\alpha$, and testing with $\alpha = 0(15^\circ)\ 345^\circ$. Having too few α values in one's test set can be quite misleading; see the results in (Hemker et al., 1983; Sonneveld et al., 1985). Another important test equation is (5.2), with the axes rotated over an angle α, resulting in

$$-(\varepsilon c^2 + s^2)\phi_{,11} - 2(\varepsilon - 1)sc\phi_{,12} - (\varepsilon s^2 + c^2)\phi_{,22} = f \qquad (5.3)$$

with $c = \cos\alpha$, $s = \sin\alpha$. This introduces the additional complication of a mixed derivative. A mixed derivative occurs in practice when a non-orthogonal coordinate mapping is used.

The fact that the coefficients in (5.1), (5.3) are constant rather than variable does not make these problems easier, but more difficult. It may easily happen that the BIM that is accelerated with CG or MG is a bad preconditioner or smoother for certain combinations of ε and α. This has more serious consequences when the unfavourable ε,α values occur throughout the domain than when these values occur only locally.

Suitable test problems with strongly discontinuous coefficients are the problems of Stone and Kershaw: see (Kettler, 1982) for a description and further references.

Whether a particular MG method will work for a particular problem may be predicted by Fourier smoothing analysis. See Kettler (1982) for an extensive catalogue of Fourier smoothing analysis results.

Sonneveld et al. (1985) report on numerical experiments, solving (5.1) and (5.3) with six methods: (1): MGD1, (2): MGD5, (3) , (4): MGD1 with horizontal or alternating zebra line Gauss-Seidel smoothing, respectively, instead of ILU, (5): CGS with ILU preconditioning, (6): CGS with IBLU preconditioning. A zebra rather than a successive ordering was used because of considerations regarding possible use of vector computers. Equation (5.1) is discretised with upwind differences so that property (3.19) holds. It is found that (2) and (6) work efficiently in all cases. The performance of the other methods deteriorates in some cases. As far as MG is concerned, the observed behaviour corresponds with smoothing analysis results, in so far as these are available. There are special cases in which methods (2) or (6) are surpassed in efficiency by one or more of the other methods, as is to be expected. For example, if ε is large in eq. (5.2) a very effective and simple preconditioner/smoother is horizontal line Gauss-Seidel (zebra or successive). If one wants to handle as large a class of problems as possible with a single autonomous (black box) code without user-provided adaptions, IBLU should be used for smoothing or preconditioning.

The experiments just mentioned were performed on a 65 x 65 grid. Computing times with CG or MG were roughly the same. We have found that as the grid becomes larger, CG starts to lag behind, in accordance with theoretical computational complexity results. But it should be remembered that CG is easier to program than MG.

6. ACKNOWLEDGEMENT

The authors are indebted to their colleague Henk van der Vorst for helpful comments.

7. REFERENCES

Alcouffe, R.E., Brandt, A., Dendy, J.E. (Jr.), Painter, J.W., (1981). "The Multi-Grid Methods for the Diffusion Equation with Strongly Discontinuous Coefficients". *SIAM J. Scient. Stat. Comp.* 2, 430-454.

Axelsson, O., (1980). "Conjugate Gradient Type Methods for Unsymmetric and Inconsistent Systems of Linear Equations". *Lin. Algebra and its Appl.* 29, 1-16.

Axelsson, O., Brinkkemper, S. and Il'in, V.P., (1983). "On some Versions of Incomplete Block-Matrix Factorization Methods". *Lin. Algebra and its Appl.* 58, 3-15.

Behie, A. and Forsyth, P., (1983). "Multigrid Solution of Three Dimensional Problems with Discontinuous Coefficients". *Appl. Math. and Comp.* 13, 229-240.

Behie, A. and Forsyth, P. (1984). "Incomplete Factorization Methods for Fully Implicit Simulation of Enhanced Oil Recovery". *SIAM J. Sci. Stat. Comput.* 5, 543-561.

Brandt, A., (1977). "Multi-Level Adaptive Solutions to Boundary-Value Problems". *Math. Comp.* **31**, 333-390.

Brandt, A., (1982). "Guide to Multigrid Development". In: Hackbusch and Trottenberg (1982), 220-312.

Brandt, A., (1984). "Multigrid Techniques: 1984 Guide, with Applications to Fluid Dynamics". Dept. of Applied Math., The Weizmann Institute of Science, Rehovot 76100, Israel.

Concus, P. and Golub, G.H., (1976). "A Generalized Conjugate Gradient Method for Nonsymmetric Systems of Linear Equations". In: R. Glowinski and J.L. Lions (eds.), Proc. of the Second Int. Symposium on Computer Methods in Applied Sciences and Engineering, Paris, 1975. *Lecture Notes in Economics and Math. Systems* **134**. *Springer-Verlag, Berlin.*

Concus, P., Golub, G.H. and Meurant, G., (1985). "Block Preconditioning for the Conjugate Gradient Method". *SIAM J. Sci. Stat. Comput.* **6**, 220-252.

Daniel, J.W., (1967). "The Conjugate Gradient Method for Linear and Nonlinear Operator Equations". *SIAM J. Numer. Anal.* **4**, 10-26.

Dendy, J.E. (Jr.), (1982). "Black Box Multigrid". *J. Comp. Phys.* **48**, 366-386.

Eisenstat, S.C., (1981). "Efficient Implementation of a Class of Preconditioned Conjugate Gradient Methods". *J. Sci. Stat. Comp.* **2**, 1-4.

Fletcher, R., (1976). "Conjugate Gradient Methods for Indefinite Systems". In: G.A. Watson (ed.), "Numerical Analysis". Proceedings, Dundee 1975. *Lecture Notes in Math.* **506**, 73-89, Springer-Verlag, Berlin.

Foerster, H. and Witsch, K. (1982). "Multigrid Software for the Solution of Elliptic Problems on Rectangular Domains: MGOO (Release 1)". In: Hackbusch and Trottenberg (1982), 427-460.

Golub, G.H. and van Loan, C.F., (1983). "Matrix Computation". *North Oxford Academic*, Oxford.

Gustafsson, I., (1978). "A Class of First Order Factorization Methods". *BIT* **18**, 142-156.

Hackbusch, W., (1982). "Multigrid Convergence Theory". In: Hackbusch and Trottenberg (1982), 177-219.

Hackbusch, W. and Trottenberg, U., (1982). "Multigrid Methods". Proceedings, Koln-Porz, 1981. *Lecture Notes in Math.* **960** Springer-Verlag, Berlin.

Hageman, L.A. and Young, D.M., (1981). "Applied Iterative Methods". Academic Press, New York.

Hemker, P.W., (1982). "A Note on Defect Correction Processes with an Approximate Inverse of Deficient Rank". *J. Comp. Appl. Math.* **8**, 137-139.

Hemker, P.W., Kettler, R., Wesseling, P. and de Zeeuw, P.M., (1983). "Multigrid Methods: Development of Fast Solvers". *Appl. Math. and Comp.* **13**, 311-326.

Hemker, P.W., Wesseling, P., de Zeeuw, P.M., (1984). "A Portable Vector Code for Autonomous Multigrid Modules". In: B. Engquist, T. Smedsaas (eds.), "PDE Software: Modules, Interfaces and Systems". *North Holland, Amsterdam*, 29-40.

Hemker, P.W. and de Zeeuw, P.M., (1985). "Some Implementations of Multigrid Linear Systems Solvers". In: Holstein, H. and Paddon, D.J. (eds.), "Multigrid Methods for Integral and Differential Equations". Clarendon Press, Oxford.

Hestenes, M.R., (1956). "The Conjugate-Gradient Method for Solving Linear Systems". Proc. Sympos. Appl. Math., vol. VI, Numerical Analysis, McGraw-Hill, New York.

Kershaw, D.S., (1978). "The Incomplete Choleski-Conjugate Gradient Method for the Iterative Solution of Systems of Linear Equations". *J. Comput. Phys.* **26**, 43-65.

Kettler, R., (1982). "Analysis and Comparison of Relaxation Schemes in Robust Multigrid and Preconditioned Conjugate Gradient Methods". In: Hackbusch and Trottenberg (1982), 502-534.

Kettler, R. and Wesseling, P., (1985). "Aspects of Multigrid Methods for Problems in Three Dimensions". Report 85-08, University of Technology, Dept. of Math. and Inf., P.O. Box 356, 2600 AJ Delft. To appear in *Appl. Math. and Comp.*

Manteuffel, T.A., (1977). "The Tchebychev Iteration for Nonsymmetric Linear Systems". *Numer. Math.* **28**, 307-327.

Manteuffel, T.A., (1978). "Adaptive Procedure for Estimating Parameters for the Nonsymmetric Tchebychev Iteration". *Numer. Math.* **31**, 183-208.

Manteuffel, T.A., (1979). "Shifted Incomplete Cholesky Factorization". In: I.S. Duff and G.W. Stewart (eds.), "Sparse Matrix Proceedings, 1978". *SIAM Publications, Philadelphia.*

McCarthy, G.J., (1983). "Investigations into the Multigrid Code MGD1". Report AERE R10889, Harwell, UK.

McCormick, S.F., (1982). "An Algebraic Interpretation of Multigrid Methods". *SIAM J. Numer. Anal.* **19**, 548-560.

Meijerink, J.A. and van der Vorst, H.A., (1977). "An Iterative Solution Method for Linear Systems of which the Coefficient Matrix is a Symmetric M-matrix". *Math. Comp.* **31**, 148-162.

Meijerink, J.A. and van der Vorst, H.A., (1981). "Guidelines for the Usage of Incomplete Decompositions in Solving Sets of Linear Equations as they occur in Practical Problems". *J. Comput. Phys.* **44**, 134-155.

Meijerink, J.A., (1983). "Iterative Methods for the Solution of Linear Equations based on Incomplete Factorization of the Matrix". Publication 643, Shell Research B.V., Kon. Shell Expl. and Prod. Lab., Rijswijk, The Netherlands, July 1983.

Meurant, G., (1984). "The Block Preconditioned Conjugate Gradient Method on Vector Computers". *BIT* **24**, 623-633.

Nowak, Z, and Wesseling P., (1984). "Multigrid Acceleration of an Iterative Method with Application to Transonic Potential Flow". In: R. Glowinski and J.L. Lions (eds.), "Computing Methods in Applied Sciences and Engineering, VI". *North Holland, Amsterdam.*

Sonneveld, P., (1984). "CGS, a Fast Lanczos-type Solver for Nonsymmetric Linear Systems". Report 84-16, University of Technology, Dept. of Math. and Inf., P.O. Box 356, 2600 AJ Delft. To appear in *SIAM J. Sci. Stat. Comp.*

Sonneveld, P., Wesseling, P. and de Zeeuw, P.M., (1985). "Multigrid and Conjugate Gradient Methods as Convergence Acceleration Techniques". In: Holstein, H. and Paddon, D.J. (eds), "Multigrid Methods for Integral and Differential Equations". Clarendon Press, Oxford.

Stüben, K. and Trottenberg, U., (1982). "Multigrid Methods: Fundamental Algorithms, Model Problem Analysis and Applications". In: Hackbusch and Trottenberg (1982), 1-176.

Stüben, K., Trottenberg, U, and Witsch, K., (1984). "Software Development Based on Multigrid Techniques". In: B. Engquist, T. Smedsaas (eds.): "PDE Software: Modules, Interfaces and Systems". *North Holland, Amsterdam.*

Underwood, R.R., (1976). "An Approximate Factorization Procedure Based on the Block Cholesky Decomposition and its Use with the Conjugate Gradient Method". Report NEDO-11386, General Electric Co., Nuclear Energy Div., San Jose, CA.

van der Sluis, A. and van der Vorst, H.A., (1985). "The Rate of Convergence of Conjugate Gradients". Dept. of Math. Utrecht University, Preprint Nr. 354. To appear in *Numer. Math.*

van der Vorst, H.A. (1981). "Iterative Solution Methods for Certain Sparse Linear Systems with a Non-Symmetric Matrix arising from PDE Problems". *J. Comput. Phys.* **44**, 1-19.

van der Vorst, H.A., (1982). "A Vectorizable Variant of some ICCG methods". *SIAM J. Sci. Stat. Comput.* **3**, 350-356.

van der Vorst, H.A., (1985). "The Performance of Fortran Implementations for Preconditioned Conjugate Gradients on Vector Computers". Report 85-09, University of Technology, Dept. of Math. and Inf., P.O. Box 356, 2600 AJ Delft.

Varga, R.S., (1962). "Matrix Iterative Analysis". *Prentice-Hall, Englewood Cliffs, New Jersey.*

Vinsome, P.K.W., (1976). "ORTHOMIN , an Iterative Method for Solving Sparse Sets of Simultaneous Linear Equations". Society of Petroleum Engineers, paper SPE 5729.

Wesseling, P. and Sonneveld, P., (1980). "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. *Lecture Notes in Math.* **771**, 543-562. Springer-Verlag, Berlin 1980.

Wesseling, P., (1982A). "Theoretical and Practical Aspects of a Multigrid Method". *SIAM J. Sci. Stat. Comp.* 3, 387-407.

Wesseling, P., (1982B). "A Robust and Efficient Multigrid Method". In: Hackbusch and Trottenberg (1982), 614-630.

Widlund, O., (1978). "A Lanczos Method for a Class of Nonsymmetric Systems of Linear Equations". *SIAM J. Numer. Anal.* **15**, 801-812.

Young, D.M., (1971). "Iterative Solution of Large Linear Systems". *Academic Press, New York.*

Young, D.M. and Jea, K.C., (1980). "Generalized Conjugate-Gradient Acceleration of Nonsymmetrizable Iterative Methods". *Lin. Algebra and its Appl.* **34**, 159-194.