MATHEMATICS OF COMPUTATION, VOLUME 33, NUMBER 147
JULY 1979, PAGES 933–952

On Some Theoretical and Practical Aspects of Multigrid Methods

By R. A. Nicolaides*

Abstract. A description and explanation of a simple multigrid algorithm for solving finite element systems is given. Numerical results from an implementation are reported for a number of elliptic equations, including cases with singular coefficients and indefinite equations. The method shows the high efficiency, essentially independent of the grid spacing, predicted by the theory.

I. Introduction. The main purpose of this report is to provide some evidence of the practical utility of multigrid methods. These methods, due originally to Federenko [11] have received development both for the finite difference case [5], [6], [7], [8], [13] and the finite element case [16], [17]. It is with the finite element case that we shall be concerned here. The theoretical predictions of the work necessary to solve any particular problem by multigrid methods are, in general, significant only in the order of magnitude sense; that is, these predictions are of the form that a given accuracy may be obtained in a number of operations proportional to N, the number of equations in a certain linear system of algebraic equations. The "constant of proportionality" is usually unknown although the factors determining it (coefficients of the partial differential operator, approximation properties of finite element trial spaces, etc.) are known. Nevertheless, it is plainly essential to determine the "constants" as far as possible, and all the more so since the $O(N)$ results alluded to above hold only for sufficiently large N.

One way of finding these unknown quantities is by means of Fourier analysis. This is done in [7]. The difficulties of this procedure are well known. Another approach is simply to solve a representative class of problems on a computer and to observe the empirical behavior of the algorithms. Naturally, this procedure is not exhaustive. In the case under discussion, however, it has some merit as we shall see later.

A secondary purpose is to offer some advice to potential users of multigrid methods. With this in mind, the next section contains some practical explanations of how the methods may be constructed and interpreted. This may be helpful as the theoretical work [16], [17] is, of necessity in view of its generality, a little abstract. The discussion is intentionally on a very simple level so that the main ideas are accessible. The rest of the paper is taken up with a discussion of numerical results obtained from

Received November 22, 1977.
AMS (MOS) subject classifications (1970). Primary 65N20.

*This report was prepared as a result of work performed under NASA Contract No. NAS1-14101 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Virginia 23665.

a relatively simple multigrid code. In Section 3 this code is outlined and in Sections 4 and 5 the results for some Poisson equations with various kinds of boundary data are given. Section 6 addresses the question of the utility of certain modifications to the basic algorithm. Sections 7 and 8 are concerned with equations with nonconstant coefficients, including discontinuous and singular cases and Section 9 deals very briefly with the indefinite case. These latter problems, whose prototype is the reduced wave equation, are not ordinarily solved iteratively. The multigrid technique makes iterative solution possible, however, and offers the usual storage economy associated with iterative methods.

II. Basic Algorithm. In this section we discuss the essential ideas of the finite element multigrid algorithm. The term *basic algorithm* refers to a stationary linear iterative method of the first degree which appears in some form in most multigrid algorithms. This algorithm may be conveniently explained in the context of a simple model problem, namely

$$\min_{u \in H_0^1(\Omega)} \int_\Omega u_x^2 + u_y^2 - 2\int_\Omega uf, \tag{2.1}$$

where $H_0^1(\Omega)$ is the familiar Sobolev space of functions with one generalized derivative and which vanish on the boundary of the bounded region Ω lying in the plane. The reader interested in a rigorous general treatment of $2m$th order boundary value problems is referred to [16].

We write (2.1) as

$$\min_{u \in H_0^1(\Omega)} a(u, u) - 2(u, f), \tag{2.2}$$

and consider the finite element solution of (2.2). Let us choose two finite element trial spaces S_1 and S_2, with S_1 containing N_1 nodal parameters and $S_2 \subset S_1$. S_1 is the usual trial space where the solution to (2.2) is sought. S_2 is an auxiliary trial space which will be required by the multigrid algorithm. S_2 is not arbitrary but is constructed in a way best made clear by an example. Thus, let Ω be the square

$$\Omega \equiv \{(x, y) | 0 < x < 1, 0 < y < 1\}.$$

We subdivide Ω by repeated halvings by lines parallel to the x and y axes. m such halvings will divide Ω into $2^m \times 2^m = 2^{2m}$ smaller elements. We shall take for S_1 the class of continuous functions which are bilinear in each element, and vanish on $\partial\Omega$. Then it is clear that $N_1 = (2^m - 1)^2$. The second trial space S_2 is naturally defined in this context as analogous to S_1, but based on $(m - 1)$ halvings of Ω. Thus, $S_1 \supset S_2$ and if N_2 denotes the number of nodal parameters of elements of S_2, $N_2 \sim \frac{1}{4}N_1$. S_2 is much smaller than S_1.

We shall adopt the following notational convention: trial functions contained in S_1 or S_2 will be written with an overbar. The corresponding vector of nodal values is then indicated by removal of the overbar. Thus, $\bar{u}$ is some trial function and u is the corresponding vector of nodal values. Returning to (2.2), the usual procedure is to substitute a general element $\bar{u}^{(1)}$ of S_1 into the variational principle to get

$$a(\bar{u}^{(1)}, \bar{u}^{(1)}) - 2(\bar{u}^{(1)}, f) \tag{2.3}$$

and then to write the conditions for a stationary point

$$K^{(1)}u^{(1)} = f^{(1)}, \tag{2.4}$$

where $K^{(1)}$ is the system or stiffness matrix for S_1, $u^{(1)}$ is an N_1-dimensional vector of nodal parameters defining the numerical solution of (2.2) and $f^{(1)}$ is computed from f and the trial functions $\phi_i \in S_1$, $i = 1, 2, \ldots, N_1$, by the formula

$$f_j^{(1)} = \int_\Omega f\phi_j, \quad j = 1, 2, \ldots, N_1. \tag{2.5}$$

The basic algorithm is for the solution of the algebraic system (2.4). Its fundamental idea is to use S_2 to construct another finite element solution to a system of the form (2.4) by exactly the same method used to derive (2.3) and (2.4) from (2.2). We cannot do this to (2.3)–(2.4) as they stand. Such an action would be tantamount simply to solving (2.2) on S_2 instead of on S_1. What we have to do is to modify (2.4) so that looking for its solution in S_2 is sensible. The mechanism for doing this is *smoothing*; in practice this is carried out by means of some relaxation. A commonly used technique is the Gauss-Seidel method. This is discussed further below. For the present we shall simply explain what the effect of smoothing is on the system (2.4).

A function $\bar{v}^{(1)}$ in S_1 will be regarded as "smooth" if it may be sensibly represented as an element of S_2. All elements in S_1 which are also in S_2 are, therefore, smooth. More generally, any element of S_1 which does not have significant fluctuations on a length scale of the order of twice the mesh length of the S_1 grid may be regarded as smooth. Let $v^{(1)}$ be an approximation to $u^{(1)}$ of (2.4) such that the residual $f^{(1)} - K^{(1)}v^{(1)} \equiv r$ is smooth. Let ϵ denote the error $u^{(1)} - v^{(1)}$ so that the error and residual are related through the equation

$$K^{(1)}\epsilon = r. \tag{2.6}$$

Since $\bar{r}$ is smooth, so is $\bar{\epsilon}$. If we could solve (2.6) for ϵ, the exact solution of (2.4) would be at hand. We cannot easily do this in general. However, we have in the case of (2.6) the vital additional information that $\bar{\epsilon}$ is capable of being reasonably represented as an element of S_2, a much smaller space than S_1. Since (2.6) is the discrete finite element system for some functional of the form (2.1) with a certain free term $\tilde{r}$, the natural suggestion is to form this functional

$$a(\bar{\epsilon}, \bar{\epsilon}) - 2(\bar{\epsilon}, \tilde{r}) \tag{2.7}$$

and attempt to minimize it on S_2, in the hope that the solution $\bar{\epsilon}^{(2)}$, a member of S_2, will be a close approximation to $\bar{\epsilon}$. Denoting the system matrix on S_2 by $K^{(2)}$, the condition for a stationary point of (2.7) is that

$$K^{(2)}\epsilon^{(2)} = r^{(2)}, \tag{2.8}$$

where $r^{(2)}$ is an N_2-dimensional vector constructed from $\widetilde{r}$ in a way we shall consider further below.

On the assumption that we are able to solve (2.8) without difficulty, we may extend $\overline{\epsilon}^{(2)}$ by the 'identity' mapping into S_1 (since $\overline{\epsilon}^{(2)} \in S_2 \subset S_1$) and denote it by $\overline{\epsilon}^{(1)}$. $\epsilon^{(1)}$ is then our approximation to ϵ and $v^{(1)} + \epsilon^{(1)}$ hopefully is near to $u^{(1)}$. This then is the essential idea behind the multigrid method: prepare the given problem in such a way that it can be represented and solved on a smaller subspace or, equivalently, on a coarser grid.

We have still to consider three things: smoothing, the question of the construction of the residual $r^{(2)}$ of (2.8) and the question of how the subproblem (2.8) is to be solved. Consider first the question of smoothing. We shall illustrate this briefly in connection with the Gauss-Seidel method. Applying this method to our system (2.4), where we assume a double indexing system for the nodes $(x_i, y_j) \equiv (ih^{(1)}, jh^{(1)})$, $h^{(1)} = (N_1 + 1)^{-1}$ and a sweeping order left to right, bottom to top, it is not difficult to verify that a given initial error ϵ transforms into ϵ' according to the rule

$$\epsilon'_{ij} = \frac{1}{8}(\epsilon'_{i-1,j-1} + \epsilon'_{i-1,j} + \epsilon'_{i-1,j+1} + \epsilon'_{i,j-1}) + \frac{1}{8}(\epsilon_{i,j+1} + \epsilon_{i+1,j-1} + \epsilon_{i+1,j} + \epsilon_{i+1,j+1}). \tag{2.9}$$

What we should notice about this and similar formulas is that the new residual at a point is a (positive) weighted average of its current neighboring values. This important property is at root a consequence of the fact that solution operators of elliptic problems are smoothing operators. The operator on the right in (2.9) is in some sense a local inverse to the elliptic operator $-\Delta$ which we are considering as our example. The property of being a smoothing operator simply means that small scale variations in the operand are eliminated (smoothed) by the operator. To help make this discussion more concrete we may note that for our model problem, three sweeps of the Gauss-Seidel method (2.9) applied to *any* initial function in S_1 are sufficient to smooth it to the point where it may be well represented by an element of S_2. It can be shown that the amount of smoothing per sweep of (2.9) is independent of $h^{(1)}$—notice that (2.9) itself is independent of $h^{(1)}$. Furthermore, these properties are still true generally speaking, for positive definite $2m$th order elliptic finite element systems [16]. This property of the smoothing behavior being independent of the grid size is the key to the remarkable convergence properties of the multigrid method. We shall have more to say about smoothing later.

Let us now turn to the question of the computation of the *reduced* residual, $r^{(2)}$ of (2.8), which is the right-hand side for the subproblem defined on S_2. This computation is effected by means of a local averaging operation on the components of r, the residual in (2.6). The precise weights to be used in this averaging process depend only on the class of piecewise polynomials used in constructing the spaces S_1 and S_2. For the model problem $r^{(2)}$ is constructed in the following way: let (x_I, y_J) be a node in the coarse (i.e., S_2) grid. Then the component $r^{(2)}_{IJ}$ is defined as

$$r^{(2)}_{I,J} = r_{I,J} + \tfrac{1}{2}(r_{I+1,J} + r_{I-1,J} + r_{I,J+1} + r_{I,J-1}) + \tfrac{1}{4}(r_{I+1,J+1} + r_{I+1,J-1} + r_{I-1,J+1} + r_{I-1,J-1}). \tag{2.10}$$

The weights in (2.10) add up to 4 because of the way finite element trial functions are scaled. The S_2 system matrix is automatically scaled in a way which corresponds with (2.10). In this sense, the method is self scaling and there is no necessity (unlike the finite difference case) for decisions by the user.

It is clear from (2.10) that $r^{(2)}$ may be obtained from r by means of a matrix multiplication

$$r^{(2)} = E^{(2)}r,$$

where $E^{(2)}$ is an $N_2 \times N_1$ matrix whose entries may be found from (2.10). In the case of more general trial functions this matrix may be easily found as the transpose of the matrix which maps each vector $u^{(2)}$ containing nodal values of an element $\bar{u}^{(2)} \in S_2$ onto the vector $u^{(1)}$ containing nodal values of $\bar{u}^{(2)}$ *regarded as an element of* S_1 [16]. We may observe that because of its definition the averaging operation is nothing more than the dual of the interpolation operation from S_2 into S_1.

Finally, we have to discuss the question of solving the reduced residual equation (2.8). Before doing this, let us point out that the general ideas so far used have a long history. In fact, as we pointed out in [15] they actually go back to Southwell [19], who used the general term "block relaxation" to describe them. Many authors have contributed to this latter circle of ideas. However, the work was always thought of as being simply a device for accelerating a relatively slowly converging iterative method. In line with this idea the reduced equation (2.8) was always constructed to be of a very small size to facilitate its solution, either by a direct method or by some other convenient technique. Federenko's outstanding idea was to observe that (2.8) is exactly another (elliptic) system of the type whose solution is being sought in the first place, and hence that it can be solved by *exactly the same method used to derive it from the original problem.* That is to say, in our context we introduce a third subspace S_3 and reduce (2.8) to a solution on it. The S_3 problem will have only about 1/16 as many unknowns as the original. The cost of making this further reduction is only a few relaxation sweeps on the S_2 grid. If the S_3 problem is still too large for convenient solution it may be further reduced at essentially negligible cost to a problem on an S_4, now containing about 1/64 of the original number of unknowns. This reduction process may be carried on until a manageable problem is obtained on some subspace S_p. Following solution on the S_p subspace, the solution is interpolated onto S_{p-1}, the new solution on S_{p-1} (possibly) smoothed to annihilate interpolation errors, then interpolated to S_{p-2} and so on. An aspect of the entire algorithm of considerable theoretical and practical importance is that it enables us to compute numerical solutions of elliptic equations with a given accuracy in essentially the minimum possible number of arithmetical operations, in the order of magnitude sense. No less important is the fact that the basic ideas are relevant not only to scalar elliptic equations, but also to systems of such equations and evolution equations. These applications have not yet been fully exploited. A large range of additional possibilities, some of them of potentially very great value are suggested for the finite difference case in [7]. There is no reason why these applications should be limited to finite differences however, and hopefully these valuable ideas will be also of service in the finite element setting.

As indicated in the introduction, we shall report below on some relatively simple multigrid computations. The computations are exclusively concerned with the basic algorithm discussed above. Regretfully, we shall not be able to consider in this report the implementation of the more powerful multigrid techniques wherein the problem to be solved is posed as that of finding an approximate solution of specified accuracy to a given differential equation. Rather, we limit the discussion to the problem of solving preassigned systems of equations which approximate particular differential equations.

III. Outline of Computations. The idea behind the computations carried out was first of all to examine the practical convergence characteristics of the basic multigrid method, especially its stability with respect to changes in implementation strategy and, secondly, to determine what variety of problems a moderately general program could effectively solve before special smoothing and other techniques become necessary. For these purposes finite element multigrid codes were written for solving second order equations of the form

$$\frac{\partial}{\partial x}\left(a(x, y)\frac{\partial u}{\partial x}\right) + \frac{\partial}{\partial y}\left(b(x, y)\frac{\partial u}{\partial y}\right) + c(x, y)u = f \tag{3.1}$$

in the rectangle $\Omega = \{(x, y) | 0 < x < x_0, 0 < y < y_0\}$ with various kinds of boundary data specified. The codes use either piecewise linear functions on triangular elements or bilinear functions on rectangular elements. In both cases the triangulations were the simplest ones, namely for the bilinear case the region Ω was dissected into n^2 identical rectangles and for the linear case, each rectangle was divided into two triangles by a diagonal with positive slope. No attempt was made to use any irregular spacing of the dissecting horizontal and vertical lines, as the effect of this can, at least for smoothly varying dissections, be simulated by variable coefficients a and b in the equation (3.1). The numerical integrations in the finite element codes were performed using the four point Gauss rule, (exact for cubics) in the bilinear case and the midpoints of the edges of the triangles as nodes in the linear case.

The calculations fall naturally into three groups. The first group consists of a fairly extensive number of tests on the Poisson equation, (3.1) with $a = b = -1$, $c = 0$. Here, in addition to Dirichlet data we have considered problems with normal derivative data given on three sides of the rectangle; in addition we tested the effect of modifying the residual reduction method in the hope of avoiding an apparently expensive computation of a residual on the finest grid at each complete cycle of the multigrid algorithm. These calculations are reported in the next three sections. The second group of calculations, much fewer in number were designed to test what loss of efficiency, if any, would be incurred when the coefficients a and b were well-behaved functions of x and y. We shall present some sample results which show that no essential loss occurs. The third group consists of some examples of solution of problems with some degree of singularity present in the coefficients. Sometimes this can cause serious trouble to a code without special relaxation methods incorporated but as we shall see this is by no means always the case.

Unless otherwise stated, each system of grids was generated by repeated dissection of the region, starting from the grid with one interior grid point. This latter was taken to be the coarsest grid. Such a choice generally avoids the need for a special solution technique on the coarsest grid, although our codes do in fact have a banded elimination solver. This is frequently useful. For example it can easily happen that the coarsest grid coefficient matrix, while positive definite and of very small size is almost singular. The relaxation method can converge too slowly to be feasible in such cases. Furthermore, as it turns out, some form of direct solution on the coarsest grid is essential for solving indefinite problems, for example these with $a = b = 1$ and $c >> 1$.

The general structure of the codes follows that of the code in [7], except that allowances are made for the fact that we have to deal with variable coefficients and different interpolation and residual reduction techniques. The total storage requirements are minimal. For example, the bilinear code uses $11N'$ storage locations for each grid where N' is the number of grid points on the grid in question. $9N'$ locations are for the coefficients; N' are for the solution on this grid, and N' are for the right-hand side. The total storage for all the grids is essentially $4/3\,\sigma_N$, where σ_N is the storage for the finest grid which contains N grid points. We allowed one vector of length N for working space. Thus the total storage requirements including coefficient storage are not larger than $16N$ locations.

Another point is that a fixed smoothing strategy is used in our codes. The meaning of this is as follows: on any grid except the coarsest and finest there are two possible occasions on which one may want to carry out smoothing (i.e. relaxation) operations. These are (a) when we want to smooth prior to forming a smaller problem on a coarser grid; this we call fine to coarse smoothing, and (b) when we want to smooth a solution which has been constructed by interpolating from a coarser grid; this is coarse to fine smoothing. Only one kind of smoothing algorithm is in the codes, namely smoothing by successive point relaxation, including Gauss-Seidel as the most important spacial case. For this smoothing method, we shall call an ordered pair of nonnegative integers (p, q) a smoothing strategy. The first integer represents the number of fine to coarse relaxation sweeps and the second the number of coarse to fine sweeps. The same pair is assigned to every grid.

In order to give a clear statement of the algorithm implemented let us introduce the following notations: the finest grid will be denoted by G_m and the successively coarser grids by $G_{m-1}, G_{m-2}, \ldots, G_1$. Let T^j_k denote an operator which transfers vectors from G_k onto G_j, where if $k < j$ the transfer operation is interpolation and if $k > j$ the operation is the reduction operation encountered several times previously. Each cycle, or iteration of the algorithm has two parts, a forward (fine to coarse) part and a return (coarse to fine) part. Denoting these two by (A) and (B), the algorithm may be described thus:

```
(A) Do steps 1 and 2 for $j = m, m - 1, \ldots, 2$.
    1. Relax on $G_j$ ($p$ sweeps); let $w^{(j)}$ and $r^{(j)}$ denote the approximation and
       residual obtained.
    2. Transfer $T_j^{j-1} r^{(j)}$ as data for $G_{j-1}$.
    3. Solve the resulting $G_1$ system for $w^{(1)}$.
(B) Do steps 4 and 5 for $k = 1, 2, \ldots, m - 1$.
    4. Form $\widetilde{w}^{(k+1)} = w^{(k+1)} + T_k^{k+1} w^{(k)}$.
    5. Relax on $G_{k+1}$ ($q$ sweeps) with $\widetilde{w}^{(k+1)}$ as initial approximation and
       denote the result by $w^{(k+1)}$.
```

$w^{(m)}$ obtained by this process is the new approximation to the solution of the algebraic system. In our codes, the starting approximations for all the intermediate relaxation solutions are taken to be zero.

All the data reported below were obtained from a time-shared CDC 6400, using an FTN FORTRAN compiler operated at the lowest level of optimization with single precision arithmetic. The timings were obtained by using the SECOND subroutine.

IV. Results for the Poisson Case I. The first topic for investigation is the question of the behavior of the method as a function of the smoothing strategy. Tables 1 and 2 contain some information along these lines, respectively, for the cases of linear and bilinear elements. Both of these tables are for a grid with $2^5 = 32$ intervals of subdivision on each side, so that there are 961 unknowns in each case. The entries in the body of the table are the number of work units expended in order to reduce the l^2 norm of the initial error by a factor 10^{-1}. A work unit here is defined as the time necessary for a relaxation sweep on the finest grid. This time was determined in all cases using the relaxation routine in our code and averaging over a large number of runs. Homogeneous data both for the right-hand side and the Dirichlet boundary condition was used. Except where stated the relaxation parameter ω was taken as 1.0. The tables show the results for a random initial error vector each of whose components is a uniformly distributed random number in the interval $[-1, 1]$. Let us explain here a characteristic difficulty of conducting these tests: although it is true that the algorithm is a linear stationary iterative method, as we shall see below it is not operated asymptotically. That is, usually a problem is solved after only a small number (say 5 for moderately sized problems) of iterations of the multigrid method. This means that the actual distribution of the initial error relative to the eigenvectors of the iteration matrix can play a significant role in the cost of a computation. It is, therefore, advisable to test the algorithm on a variety of initial approximations of differing characters. However, even this cannot guarantee that the worst possible case has been tested. On the other hand it is not necessarily true that the worst case will always occur in a practical situation. Although we shall not report extensively on the computations with initial errors other than the random one mentioned above, other initial errors have in fact been tried. We believe that our reported figures are a reliable indication of what can be achieved in less academic situations. The feature of nonasymptotic operation of the method requires us also to examine the variation in cost according to the number of iterations carried out. This is the reason for the three figures given for each (p, q) strategy. Reading left to right the costs are, respectively, for 1, 3, and 5 iterations of the algorithm.

q \ p	1	2	3	4	5	LINEAR
5	6.6 7.6 8.3	5.1 6.0 6.4	4.7 5.7 6.2	4.5 5.5 6.0	4.6 5.9 6.3	ELEMENTS
4	6.1 7.1 7.6	4.5 5.5 6.0	4.4 5.5 5.8	4.7 5.7 6.0	4.6 5.7 6.1	32 × 32 GRID UNIFORM
3	5.7 6.5 7.0	4.3 5.3 5.6	4.3 5.2 5.5	4.3 5.3 5.5	4.5 5.5 6.0	RANDOM INITIAL ERROR
2	5.3 6.1 6.5	4.2 5.0 5.4	4.0 5.0 5.3	4.0 5.0 5.4	4.3 5.4 5.7	1 3 5
1	4.9 5.5 6.0	3.9 4.6 4.9	3.7 4.6 5.0	3.9 4.8 5.2	4.3 5.1 5.5	ITERATIONS
0	4.6 5.4 5.6	3.7 4.4 4.7	3.7 4.6 4.8	4.0 4.9 5.2	4.0 4.9 5.2	WORK UNITS BASED ON 107 × 10^{-3} sec/SOR SWEEP

TABLE 1. Strategies

q \ p	1	2	3	4	5	BILINEAR
5	5.2 5.3 5.5	4.0 4.0 3.9	3.9 4.4 4.4	3.7 4.2 4.2	3.9 4.4 4.4	ELEMENTS
4	5.0 5.1 5.3	3.7 3.7 3.6	3.5 4.0 4.0	3.9 4.2 4.1	4.0 4.4 4.4	32 × 32 GRID UNIFORM
3	4.4 4.6 4.7	3.6 3.5 3.5	3.5 4.0 4.0	3.7 4.0 4.0	3.9 4.3 4.3	RANDOM INITIAL ERROR
2	4.3 4.4 4.5	3.3 3.4 3.3	3.4 3.7 3.7	3.4 3.8 3.9	3.8 4.2 5.6	1 3 5
1	3.6 3.4 4.2	3.1 3.1 3.1	3.0 3.6 3.5	4.0 3.7 3.7	3.6 4.0 4.0	ITERATIONS
0	3.6 3.7 3.9	3.0 3.0 3.2	3.1 3.4 3.4	3.3 3.7 3.6	3.5 4.0 4.0	WORK UNITS BASED ON 135 × 10^{-3} sec per SOR ITERATION

TABLE 2. Strategies

The tables show in both cases a high degree of robustness in the algorithm, indicating, respectively, costs ranging between say 4–8 work units and 3–5½ work units per 10^{-1} reduction in the error (i.e., per digit) over the range of strategies (p, q), $0 \leq p, q \leq 5$. (We shall frequently approximate the tabulated figures in this way. As a justification we may note that the starting errors are "random" and not, strictly speaking, reproducible and also that slight deviations in run times can occur on time-shared computers.) The ranges are even smaller if the unreasonable strategies of the form (p, q), $q >> p$ are eliminated. A clear feature in both cases is the increase in cost with the number of iterations. This phenomenon is commonly observed with iterative methods and apparently the multigrid method—at least when operated in the stationary mode we are using here—suffers from this defect. We have computed the costs for larger numbers of iterations and found that the figures for 5 iterations whilst not maximal are nevertheless essentially reliable. In practice, 5 iterations would give us several correct digits so that the question of doing large numbers of iterations is unlikely to be of practical interest.

Another interesting point is that the best strategies in both cases are along the bottom row of the tables, where $q \equiv 0$. This means that the greatest efficiency is obtained when *no* smoothing is carried out following the transfers from coarse to fine grids when the starting error is random. However, it is essential that we point out that other calculations we have carried out but not reported here have shown that when the starting error is very smooth, for example a constant, the greatest efficiency is obtained when $q = 1$. However, the loss in using $q = 0$ in this case is minor and, in fact, is comparable with the loss that is incurred if we use $q = 1$ in the case of the random starting error—the second row up in Tables 1 and 2. Notice that a best overall strategy indicated by the tables is the (2, 0) strategy in both cases. One further point to notice is that, although the bilinear elements require only about 3 work units per digit against about 5 for the linear elements, the latter require less time for a fine grid relaxation sweep because of the fact that in general only 7 operations are required at each point instead of 9 in the bilinear case.

GRID SIZE	BEST STRATEGY	COST/ 10^{-1} R	ITNS/ 10^{-1} REDN	ERROR TYPE	LINEAR ELEMENTS
8 × 8	(4,0)	4.2	0.7	R	R = RANDOM
	(2,3)	4.7	1.1	S	S = SMOOTH STARTING ERRORS
16 × 16	(3,0)	4.5	0.9	R	FIGURES BASED
	(1,1)	4.2	1.5	S	ON 3 ITNS
32 × 32	(2,0)	4.4	1.3	R	SOR SWEEP TIMES
	(2,2)	3.9	1.0	S	
64 × 64	(3,0)	4.6	1.0	R	5×10^{-3}, 24×10^{-3}
	(2,3)	3.9	0.9	S	107×10^{-3}, 425×10^{-3} sec

TABLE 3. Best strategies

GRID SIZE	BEST STRATEGY	COST/ 10^{-1} R	ITNS/ 10^{-1} REDN	ERROR TYPE	BILINEAR ELEMENTS
8 × 8	(2,0)	3.0	0.7	R	STARTING ERRORS
	(1,1)	3.2	1.0	S	R = RANDOM
16 × 16	(2,0)	2.9	0.8	R	S = SMOOTH
	(1,1)	2.9	1.0	S	FIGURES BASED ON 3 ITNS
32 × 32	(2,0)	2.9	0.8	R	SOR SWEEP TIMES
	(1,1)	2.8	1.0	S	6×10^{-3}, 31×10^{-3}
64 × 64	(2,0)	3.1	0.8	R	135×10^{-3}, 540×10^{-3} sec
	(1,1)	2.8	0.9	S	

TABLE 4. Best strategies

Consider next Tables 3 and 4. These tables are intended to indicate the best strategies found on the sequence of grids shown, so that some idea of the dependence upon the grid size may be seen. These tables contain data for the case of a smooth (actually constant) as well as a random initial error. We have also included additional information, such as that shown in the fourth columns of the tables in order to convey some idea of the time required to solve a given problem. Thus, for example in the bilinear case 0.8–1.0 multigrid cycles will generate about one correct digit in the numerical solution, and (3–4) work units will be required to do the calculations. The most important fact to be learned from Tables 3 and 4, however, concern the dependence on the grid size. It is clear that for all practical purposes the method has convergence properties independent of the grid spacing. It appears that whatever the mesh spacing, in the bilinear case we can obtain each new digit at a cost of about 3 work units, and in the linear case at a cost of about 4–5 work units. Roughly speaking, this means a cost of about 30–35 N multiplications to obtain each new digit in either case, where N is the number of unknowns. Furthermore, comparing the linear case with the optimum SOR times, the multigrid method is 4 times faster for the 32 × 32 interval grid for a given accuracy and 8 times faster for the 64 × 64 interval grid. On a 128 × 128 interval grid the multigrid method would be 16 times faster than SOR. The slight discrepancies between Tables 3 and 4 and Tables 1 and 2 are caused mainly by the fact that different random starting errors were used in the various cases. Let us also note that the cost figures for the case of fewer multigrid iterations are marginally lower on average and those for more iterations marginally higher on average than those reported in the tables. No attempt has been made to optimize our figures. The codes were all run with the lowest level of compiler optimization and, as we said before, with the relaxation parameter $\omega = 1$. It would be possible to obtain further speed up factors of about 10%–20% by adjustment of the various parameters. However, our main concern here is with the average behavior of the method over a wide class of problems, and not with optimizing it for any specific application. We regard the generality of the algorithm as one of its most significant attributes.

A somewhat disturbing feature of Table 3 is the lack of regularity in the best strategy in the various cases. To show that this is not too serious a problem we present Table 5 which contains the results of using in all cases the strategy (3, 0) on ran-

GRID SIZE	BEST STRATEGY	COST/ 10^{-1} R	ITNS/ 10^{-1} REDN	ERROR TYPE	LINEAR ELEMENTS
8 × 8	(3,0)	5.1	1.0	R	STARTING ERRORS
	(2,2)	4.9	1.1	S	R = RANDOM
16 × 16	(3,0)	4.5	1.0	R	S = SMOOTH
	(2,2)	4.2	1.0	S	FIGURES BASED
32 × 32	(3,0)	4.6	1.0	R	ON 3 ITNS
	(2,2)	3.9	1.0	S	SOR SWEEP TIMES
64 × 64	(3,0)	4.6	1.0	R	5×10^{-3}, 24×10^{-3}
	(2,2)	4.0	1.0	S	107×10^{-3}, 425×10^{-3} sec

TABLE 5. Suboptimal strategies

dom errors and (2, 2) on smooth errors. The solution costs are seen to be entirely comparable with those of using the optimal strategies. Notice that Tables 3, 4, and 5 show consistently lower costs for solving problems with a smooth starting error. It follows that such starting errors should be arranged for wherever possible. However, this cannot always be arranged, particularly when a sequence of elliptic problems is to be solved in the course of solution of a time dependent problem, an eigenvalue problem or similar situations.

To summarize our conclusions so far we may state that 30–35 N multiplications per digit is a reasonable cost to allow for the solution of Poisson's equation in a square using the algorithm described. As an empirical observation, it is also suggested that smoothing strategies of the form $(p_1, 0)$ and (p_2, p_2) for small integers p_1 and p_2 are likely to be of greatest practical interest.

V. Results for Poisson's Case II. In this section we shall consider the same equation as in Section 4, but with the boundary conditions

$$\begin{aligned} u = 0, \quad x = 0, \quad 0 \leqslant y \leqslant 1, \\ \partial u/\partial n = 0, \quad \text{elsewhere on } \partial\Omega, \end{aligned} \tag{5.1}$$

where $\partial u/\partial n$ denotes differentiation in the direction of the outward pointing normal to $\partial\Omega$. Equation (5.1) was treated in the usual way as a natural boundary condition.

The results of some calculations are summarized in Tables 6 and 7.

LINEAR ELEMENTS, 32 × 32 GRID, 3 ITERATIONS										
ERR \ ω	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7
RANDOM	6.6	5.9	5.6	5.9	6.4	7.1	8.3	10.3	-	-
SMOOTH	-	-	12.1	9.8	8.3	6.6	5.1	4.0	4.4	4.8

TABLE 6. Costs as function of ω

BILINEAR ELEMENTS, 32 × 32 GRID, 3 ITERATIONS										
ERR \ ω	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7
RANDOM	4.5	4.1	4.0	4.4	5.1	5.4	6.2	-	-	-
SMOOTH	-	-	8.3	6.8	5.8	5.1	4.1	4.2	4.7	5.6

TABLE 7. Costs as function of ω

These tables are self-explanatory. The smooth starting error referred to is a constant vector. The figures for the work units are those for the best strategy found in every case. As the tables show, $\omega = 1$ is a satisfactory relaxation parameter for either element type with a random starting error. In this case, the best strategy for both types of elements was stable for large variations in ω about $\omega = 1$, and was found to be the (2, 0) strategy. The computation costs are a little higher than for the Dirichlet problem. A heuristic explanation of this fact is that the discrete system of equations now contains a number of first order difference equations associated with the Neumann condition, in addition to the second order equations associated with the interior points. These first order equations necessarily exert a destabilizing effect on the algorithm, and so increase the solution costs. It is worth pointing out here that an analogous loss of convergence speed occurs with SOR and other relaxation methods when applied to problems involving Neumann data.

A more striking fact is the behavior in each case associated with the smooth error. Here, it turned out to be essential to overrelax in order to make the solution costs comparable with those of the Dirichlet case. A full explanation of this phenomenon would probably involve the fact that an error vector e, equal to 1 in every component, has a residual which vanishes everywhere except near the left boundary of Ω, so that the residual equation in a sense contains relatively poor information. We shall not go into this any further here. Let us instead note that using the higher values of ω, say $\omega = 1.5$, enables the computation costs to once again be made comparable with those for the Dirichlet case. As with the random starting error, the best strategy for both kinds of elements was very stable with respect to large changes in ω about its best value, and was in both cases the (1, 1) strategy. Also, the costs for a given ω were found to behave similarly to those for the Dirichlet case under variations in the smoothing strategy. These computational costs are all essentially grid independent.

VI. Modified Residual Reduction. A major contribution to the solution cost in using the multigrid method comes from the evaluation of the residual prior to its reduction onto a coarser grid. It is, therefore, natural to try to avoid this calculation. Clearly, the best we can hope to do is to reduce the cost from that of a fine grid calculation to that of an adjacent coarse grid calculation. The most obvious method is simply to calculate the residual at the coarse grid points only and to use the values so obtained as the data for the reduced problem. Apparently, to do anything else in the finite difference case is wasteful [7, Eq. A.12a]. In the terminology of [7] this reduction method is called injection of the residuals.

The question that arises is whether or not the injection method can yield comparable accuracy for the smaller amount of work that it requires. On the basis of a considerable number of calculations, our conclusions about using injection in the finite element ease are as follows: firstly, in no case were we ever able to obtain any essential reduction in the computation cost, although in the Dirichlet cases it is usually possible to obtain similar cost figures for the algorithms using injection on the one hand and the weighted average reduction of previous sections on the other. In the case of the mixed problem of Section 4 the injection method is substantially more expensive to compute with. Secondly, there is a loss of robustness with the injection method. For example, use of some plausible smoothing strategies can actually cause divergence to occur. We have yet to see divergence occur when solving a positive definite problem using the full algorithm. Tables 8 and 9 contain some results illustrating these points. For the mixed problem, results are not so bad when the starting error is smooth (not shown in the tables) but are still 50% or so more than with the full algorithm. Notice also in the mixed problem that there is a loss of stability, in the sense that small changes in ω can lead to relatively large changes in the costs.

q \ p	1	2	3	4	5	6	LINEAR ELEMS.
5	*	9.0	7.5	5.6	5.2	5.0	DIRICHLET CASE
4	*	8.1	6.4	5.4	5.1	5.0	32 × 32 GRID
3	*	9.4	6.7	5.5	4.9	4.8	3 ITNS. RANDOM
2	*	7.3	5.8	4.7	4.6	4.4	INITIAL ERROR
1	142	6.1	5.8	4.7	4.4	4.5	* MEANS
0	*	7.0	5.4	4.6	5.0	4.8	DIVERGED

TABLE 8. Strategies with injection ($\omega = 1.0$)

32 × 32 GRID, 3 ITNS, R. INITIAL ERROR, MIXED BDRY DATA							
ELEM \ ω	0.6	0.7	0.8	0.9	1.0	1.1	1.2
LINEAR	15.9	10.5	7.9	9.8	12.5	17.6	-
BILINEAR	-	10.4	8.2	7.7	7.8	8.3	11.5

TABLE 9. Costs as function of ω, with injection

For these reasons, we do not recommend the use of injection as a general purpose approach. On the other hand, it may well be that in the context of optimizing an algorithm for a specific problem, the storage saved (possibly as much as 1 grid length vector for the finest grid) would outweigh the disadvantages.

VII. Variable Coefficients. In this short section we shall show some solution costs for equations with smooth coefficients. All the examples shown below were computed using a random initial error, a (2, 0) smoothing strategy and three multigrid iterations.

COEFFICIENTS	COST/10^{-1} REDN.	GRID
$a = b = [1 + \frac{1}{2}(x^4 - y^4)]^2$	3.3	64 × 64
$a = b = [1 + \sin \frac{1}{2}\pi(x + y)]^2$	3.3	64 × 64
$a = b = [2 + \tanh\ 4(x + y - 1)]^2$	3.3	64 × 64
$a = b = [1 + 4\,\lvert x - \frac{1}{2}\rvert]^2$	3.3	64 × 64
$a = b = e^{xy} \sin(\sqrt{x} + y^2)$	3.0	32 × 32
$a = 1 \quad b = e^{xy} \sin(\sqrt{x} + y^2)$	3.4	32 × 32

TABLE 10. Smooth coefficients

The first four of these examples are taken from [10]. No extensive comment on Table 10 is needed. It is clear that solving this type of problem is essentially equivalent in terms of cost to solving Poisson's equation. As with the latter case, the costs do not depend on the grid size to any great extent.

VIII. Discontinuous and Singular Coefficients. Here we present some calculations carried out for equations with various kinds of singular coefficients. All the data shown are for bilinear elements with a random initial error, the (2, 0) smoothing strategy and three multigrid iterations.

Example 1. Here we take

$$a = b = |\sin kx \sin ky|, \qquad c = 0, \qquad f = 0,$$

with homogeneous Dirichlet data on $\partial\Omega$; k is a parameter. The results given in Table 11 are for the 32 × 32 and the 64 × 64 interval grids. Costs are the usual work units needed for a 10^{-1} reduction in the initial error. Note that use of $f = 0$ and homogeneous boundary data avoids the potential over-determination of this problem.

k / GRID	2	4	8	16	32
32 × 32	4.7	5.2	5.4	5.9	5.0
64 × 64	4.5	4.8	5.8	6.3	6.1

TABLE 11. Costs

The vanishing of the coefficient a inside the region Ω when $k > \pi$ means that the quadratic form associated with this problem is not uniformly positive definite. For this reason the theory in [16] is not valid here. Nevertheless, the results show that the method is not very grid dependent; and that the computation costs are something less than double what they are for the Poisson case.

A point worth observing is that for the larger values of k the coarser grids are not such as to permit approximation of the differential operator in the usual senses. They do, however, appear to be able to provide approximation properties appropriate to the multigrid method. This is because the discrete operators associated with the coarser grids contain averaged information about the differential operator which is made use of by the multigrid algorithm. To see whether increasing the fineness of the coarsest grid made any real difference to the solution costs we calculated some additional examples. Table 10 shows some calculations on the 32 × 32 interval grid where the coarsest grid was taken as the 4 × 4 interval grid. The coarsest grid problem was solved using the band solver in the code. There appears to be a slight systematic improvement compared with Table 11 although in other examples not shown this improvement is not always maintained.

k	2	4	8	16	32
32 × 32	4.7	5.1	5.0	5.4	4.8

TABLE 12. Costs

Example 2 [10]. To show that singular coefficients do not always lead to greatly increased computation costs we mention the following example in which $a = b$, $c = f = 0$ and

$$a = \begin{cases} 1.0, & 0 \leqslant x \leqslant 0.5, \\ 9.0, & 0.5 < x \leqslant 1.0, \end{cases}$$

are solved under the same conditions as Example 1. Here we found on the 64×64 interval grid the cost figure 3.4 units. Similar costs are observed on other grids.

Example 3. This is another example with discontinuous coefficients, although with a positive definite energy functional. Again we take $a = b$, $c = 0$ and $f = 0$ with Dirichlet data. To define the coefficient a, the region is divided into $2^k \times 2^k$ equal squares by lines parallel to the x and y axes. a is then defined checkerboard fashion as equal to ϵ (a parameter) and 1 in alternate small squares. Thus, for $k = 1$ there are four squares and reading left to right, top to bottom, a is alternately ϵ, 1, 1, ϵ. We have computed these examples for $k = 1, 2$, and 3, the latter case containing 64 changes in the coefficient a. More complex problems of this type are of considerable practical interest [1]. The examples reported below are all for the 32×32 subinterval grid using bilinear elements, a random starting error, and three multigrid iterations as previously. In these examples, the coarsest grid was chosen to coincide with the subdivision defining the coefficient a. The coarsest grid problems were solved directly by the banded elimination solver. The results are reported in Table 13 for the cases $k = 1, 2$, and 3.

$k \backslash \varepsilon$	10^{-1}	10^{-2}	10^{-3}	10^{-4}	10^{-5}
1	3.6	4.2	4.1	4.1	4.1
2	4.7	5.9	6.1	6.0	6.2
3	4.0	4.9	5.1	5.1	5.1

TABLE 13. Costs

The anomalous behavior in the second row is not fully explained but may be caused by a particularly difficult initial error. The costs seem to be relatively independent of ϵ, although rising with k. They are on average considerably less than twice the costs for the Poisson case.

Example 4. The final example in this section is for the operator

$$Lu \equiv \epsilon u_{xx} + u_{yy}, \tag{8.1}$$

where ϵ is a positive constant. This case, which embodies certain characteristics of the transonic flow small disturbance equation, is known to have a slow multigrid convergence rate when $\epsilon \ll 1$. A remedy is also known [14], [9], namely, to use line relaxation along lines $x =$ constant. Equations with a form of which (8.1) is a simple model arise in many applications, particularly in nonlinear problems where the term(s) multiplied by the small factor ϵ may not be a priori known.

The idea of our test here was to determine what coefficient ratio 1: ϵ can be handled without too much loss of efficiency by point relaxation. This is important to us inasmuch as it delimits the area of usefulness of our general purpose approach—that is, we hope to learn what can be done before special relaxation techniques become necessary.

The calculations in Table 14 were obtained on the 32 × 32 interval grid in the same circumstances as the previous examples of this section.

ε	1.0	.75	.5	.25	.1
32 × 32	3.2	3.4	4.0	5.3	8.2

TABLE 14. Costs

These data were obtained using a random initial error; they indicate that coefficient ratios of the order 5 : 1 can be dealt with reasonably well.

A more stringent test suggested by the Fourier analysis of the smoothing properties of the point relaxation method [7] is to use a special starting error, namely the following: we use a vector constant on the lines x = constant and with the periodic variation in the x direction 0, + 1, 0, − 1. The data for this example on two different grids are shown in Table 15 below:

ε / GRID	1	.75	.5	.25	.1
32 × 32	2.5	2.9	4.0	7.1	16.3
64 × 64	2.6	3.0	4.3	7.9	17.7

TABLE 15. Costs

For the smaller values of ϵ, these costs are considerably worse than those of Table 14. However, they can be reduced by a change in the smoothing strategy to (3, 0) and some slight overrelaxation $\omega = 1.2$ as shown in Table 16 for the 64 × 64 grid.

ε	1.0	.75	.5	.25	.1
32 x 32	3.0	3.0	3.2	4.2	10.9

TABLE 16. Costs

It appears that with this modification (not the optimal one by any means) the figure 5 : 1 for the coefficient ratio is still realistic. In those applications where the coefficient ratio is considerably larger than 5 : 1 special relaxation methods become necessary. With the use of these methods the cost figures once again become comparable to those of the Poisson case.

IX. Indefinite Case. in [17] we prove multigrid convergence results for the indefinite case paralleling the results of the definite case in [16]. Generally speaking, indefinite problems, of which the Helmholtz equation

$$u_{xx} + u_{yy} + c^2u = f \qquad (c^2 >> 1) \tag{9.1}$$

is a prime example, are harder to solve iteratively than definite problems such as the ones considered in previous sections. The basic reason for this is that the standard iterative methods, such as successive relaxation, the Jacobi method, and methods using orthogonal polynomials for acceleration purposes (including the conjugate gradient techniques) do not converge when the difference operator has both positive and negative eigenvalues. It is possible to derive a positive definite operator by squaring the indefinite one, but the deficiencies of this approach are well known. It is, therefore, of some interest to note [17] that the multigrid method not only converges, but converges optimally (in the order of magnitude sense) when applied to a large class of indefinite problems which includes (9.1) as a special case.

On the other hand, we must point out here that very substantial difficulties arise when, e.g. we try to solve (9.1) with very large c^2. Quite apart from the difficulties of representing the solution in this case, the multigrid method encounters its own difficulties which we may sum up as follows: for convergence the coarsest grid must not be too coarse. This does not conflict with the theoretical results as these are of an asymptotic nature, valid in the limit as the mesh size tends to zero. However, in some settings it seems that we can have a situation where we cannot use a conveniently coarse grid. A conclusive resolution of this difficulty is not yet known to the author.

For the case of relatively small c^2 however, which embraces many (if not most) cases of practical interest, our tests so far indicate that the usual rapid convergence is achievable. We envisage here a situation where c^2 is such that a small number (e.g. 6) of eigenvalues of (9.1) are positive, while the rest are negative. We shall report on some computations along these lines, together with implementation guidelines in [12]. As an example of what can be achieved, we wish to mention one simple computation, taken from [12]. This is a model of a problem in duct acoustics and, in fact, is similar to the three-sided Neumann data problem considered in Section 5. We take the equation

$$\Delta u + 4.0u = f \tag{9.2}$$

on the 32×32 interval grid, and specify the same data as in Section 5, (5.1). The operator in (9.2) with this data has one positive eigenvalue, λ_1, where

$$\lambda_1 = 4.0 - \pi^2/4.$$

The other eigenvalues are negative. Starting with a fairly smooth error, the (2, 0) smoothing strategy, $\omega = 1.0$ and solving exactly on the 4×4 interval grid, we find the cost figure 4.1 work units per 10^{-1} reduction in the initial error, for bilinear elements. Thus, there are grounds for expecting that at least moderately indefinite problems can be efficiently solved.

X. Concluding Remarks. Although we have considered only the simplest class of problems in the previous sections it is reasonable to suppose that the basic multigrid algorithm will perform similarly well in more general settings. In the author's opinion, the basic algorithm is a highly efficient and versatile tool for solving positive definite and moderately indefinite problems. The situation is not so unequivocal in the indefinite case for the reasons previously indicated. Here we feel that for the exceptionally difficult problems, additional studies, both theoretical and practical, will be required.

One major problem we have not mentioned so far is that of defining the nested sequence of grids on an arbitrary region. Our experience with this is that in any particular case this is relatively easy to do. However, we encountered difficulty in constructing a general code to handle an arbitrary region in a way that did not require a large amount of extra input data—something to be avoided at all costs—from the user. One possible approach is to transform the physical region onto a union of simpler ones where nested grids are easily defined. Unfortunately, *efficient* methods for numerically carrying out such transformations have not so far been extensively studied.

Another topic of great potential value is that of adaptive computation. In the multigrid context, where adaptation is especially natural, this application has been investigated in [7], [8]. Adaptive finite element computation has been pioneered in [2], [3], [4] although not using multigrid ideas in the sense we are considering them here. There is great scope for applying the multigrid methods in this context.

To conclude, we hope that our results have shed some light on implementation questions arising from the multigrid method. Further computations are given in [18] and program listings are obtainable from the present author.

Acknowledgments. I would like to express my thanks to Mr. Craig Poling of the College of William and Mary for his diligence and patience in obtaining the numerical data used to compile this report. Mr. Poling's Master's thesis [18] contains additional details of the computations reported here and many further computations.

Department of Mathematics
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

1. I. BABUŠKA, "Homogenization and its application," Mathematical and Computational Problems, *Numerical Solution of Partial Differential Equations,* III (*SYNSPADE* 1975), Academic Press, New York, 1976, pp. 89–116.

2. I. BABUŠKA, "The self-adaptive approach in the finite element method," *Mathematics of Finite Elements and Applications* (J. R. Whiteman, Ed.), Academic Press, London, pp. 125–143.

3. I. BABUŠKA & W. RHEINBOLDT, *Computational Aspects of Finite Element Analysis,* Computer Science Technical Report TR-518, University of Maryland, April, 1977, pp. 1–31.

4. I. BABUŠKA & W. RHEINBOLDT, *Error Estimates for Adaptive Finite Element Computations,* Inst. Phys. and Tech, Technical Note BN-854, University of Maryland, May, 1977, pp. 1–41.

5. N. S. BAKHVALOV, "On the convergence of a relaxation method under natural constraints on an elliptic operator," *Ž. Vyčisl. Mat. i Mat. Fiz.,* v. 6, 1966, pp. 861–883. (Russian)

6. A. BRANDT, *Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems,* Proc. 3rd Internat. Conf. on Numerical Methods Fluid Mechanics (Paris, 1972); Lecture Notes in Physics, Vol. 18, Springer-Verlag, Berlin, 1972, pp. 82–89.

7. A. BRANDT, "Multi-level adaptive solution to boundary value problems," *Math. Comp.*, v. 31, 1977, pp. 333–391.

8. A. BRANDT, *Multi-Level Adaptive Techniques (MLAT): Ideas and Software*, Proc. Conf. Mathematical Software; MRC, Wisconsin, 1977.

9. A. BRANDT & J. R. SOUTH, JR., *Application of a Multi-Level Grid Method to Transonic Flow Calculations*, ICASE Report No. 76–8, 1976.

10. P. CONCUS & G. GOLUB, "Use of fast direct methods for the efficient numerical solution of nonseparable elliptic equations," *SIAM J. Numer. Anal.*, v. 10, 1973, pp. 1103–1120.

11. R. P. FEDERENKO, "The speed of convergence of an iteration process," *Ž. Vyčisl. Mat. i Mat. Fiz.*, v. 4, 1964, pp. 559–564. (Russian)

12. M. D. GUNZBURGER & R. A. NICOLAIDES. (In preparation.)

13. W. HACKBUSCH, "A fast iterative method for solving Poisson's equation in a general region," *Numerical Treatment of Differential Equations* (R. Bulirsch et al., Eds.), Lecture Notes in Math., Springer-Verlag, Berlin, 1977.

14. A. JAMESON, Personal communication.

15. R. A. NICOLAIDES, "On multiple grid and related techniques for solving discrete elliptic systems," *J. Computational Phys.*, v. 19, 1975, pp. 418–431.

16. R. A. NICOLAIDES, "On the l^2 convergence of an algorithm for solving finite element equations," *Math. Comp.*, v. 31, 1977, pp. 892–906.

17. R. A. NICOLAIDES, "On multigrid convergence in the indefinite case," *Math. Comp.*, v. 32, 1978, pp. 1082–1086.

18. T. CRAIG POLING, M.A. Thesis, College of William and Mary, Williamsburg, Virginia, 1977.

19. R. V. SOUTHWELL, *Relaxation Methods in Theoretical Physics*, Clarendon Press, Oxford, 1946.