MATHEMATICS OF COMPUTATION
VOLUME 41, NUMBER 163
JULY 1983, PAGES 43-62

Unigrid for Multigrid Simulation

By S. F. McCormick and J. W. Ruge

Abstract. This paper develops *unigrid* which is an algorithm that results from interpreting multigrid as it directly affects the fine grid approximation. Unigrid is theoretically equivalent to multigrid under certain assumptions, yet has different computational characteristics. Though it is generally less efficient, unigrid is very useful as a software tool for testing the feasibility of applying multigrid to a given problem. This is illustrated with several numerical examples.

1. Introduction. The subject of this paper is *unigrid*, a method designed primarily for solving the equations arising from discretization of partial differential boundary value problems. The method is closely related and, under certain conditions, theoretically equivalent to multigrid, although all of the unigrid computations are performed on a single grid, as the name implies. Unigrid is generally more computationally costly than multigrid, yet its simplicity is reflected in its relative ease of code implementation and use with existing applications software. This enables unigrid to act effectively as a software tool to simulate the multigrid approach that requires a more intensive design effort.

Because of its conceptual simplicity, in a *limited* way the ideas basic to unigrid can be understood independently from those of multigrid. In fact, unigrid is described below as a directional iteration method and can therefore be considered from a purely algebraic point of view. However, the ability to use unigrid effectively depends on an understanding of the discretization and relaxation principles that are the foundation for multigrid. Simplicity of description should not compromise these principles, so it is advised that the reader have a basic understanding of multigrid (cf. [1]).

On the other hand, for the reader that does not have this understanding, and for general concreteness, the next section contains a simplified description of unigrid applied to a one-dimensional model problem, However, the remaining sections assume familiarity with the concepts and notation of multigrid and include the general development of unigrid (Section 3); its relationship to multigrid (Section 4); some theory (Section 5); descriptions of several versions of unigrid together with code and output listings for a few that were implemented in BASIC on a Hewlett-Packard 9845B (Section 6); and the use of unigrid illustrated with some experiments on irregular boundaries and nonlinearities (Section 7).

Received July 10, 1981; revised January 8, 1982.
1980 *Mathematics Subject Classification.* Primary 65N99; Secondary 65N20, 65F10.

0025-5718/83/0000-1239/$05.50

2. A Simple Example. To describe unigrid in its simplest form, consider the two-point boundary value problem

$$-y''(x) = f(x), \qquad x \in [0,1], \tag{2.1}$$
$$y(0) = y(1) = 0.$$

Assuming that $h = 2^{-m}$ for some positive integer m, let $n = 2^m - 1$, define the grid points by $x_k = kh$, $1 \leq k \leq n$, and suppose that

$$A^h U^h = f^h, \qquad U^h \in R^n, \tag{2.2}$$

denotes a suitable discrete approximation to (2.1). For example, the components of f^h might correspond to f evaluated at the grid points, and A^h might denote the usual n by n tridiagonal central difference matrix with diagonal entries $2h^{-2}$ and off-diagonal entries $-h^{-2}$.

In what follows, upper and lower case letters will denote, respectively, exact and approximate solutions of (2.2).

The kth step of one sweep of Gauss-Seidel relaxation can now be written as

$$u^h \leftarrow u^h - \frac{\langle r^h, e_k^h \rangle}{\langle A^h e_k^h, e_k^h \rangle} e_k^h. \tag{2.3}$$

Here, e_k^h is the kth column of the n by n identity matrix, the residual is given by $r^h = A^h u^h - f^h$, and the left arrow denotes replacement, thereby avoiding iteration subscripts.

Heuristically speaking, the main difficulty with Gauss-Seidel applied to (2.2) stems from its restriction to the use of local data. More specifically, information (i.e., residual error) at grid points distant from x_k takes many sweeps to properly affect u_k^h. Thus, the "spiked" grid functions e_k^h depicted in Figure 1 do little to provide communication between widely separated grid points.

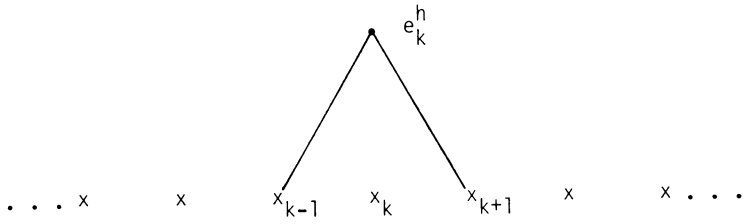

FIGURE 1

This interpretation suggests the use of "broader-based" directions such as those defined recursively by

$$d_k^H = \begin{cases} .5d_{2k-1}^{H/2} + d_{2k}^{H/2} + .5d_{2k+1}^{H/2}, & H = 2^i,\ 1 \leq i \leq m-1, \\ & \qquad 1 \leq k \leq 2^i - 1, \\ e_k^h, & H = h,\ 1 \leq k \leq n. \end{cases} \tag{2.4}$$

These directions are illustrated in Figure 2 for the case $m = 3$. Note, in general, that these directions for level H are simply the grid functions e_k^H (defined on grid points $x_{2^i k}^h$) linearly interpolated to grid h.

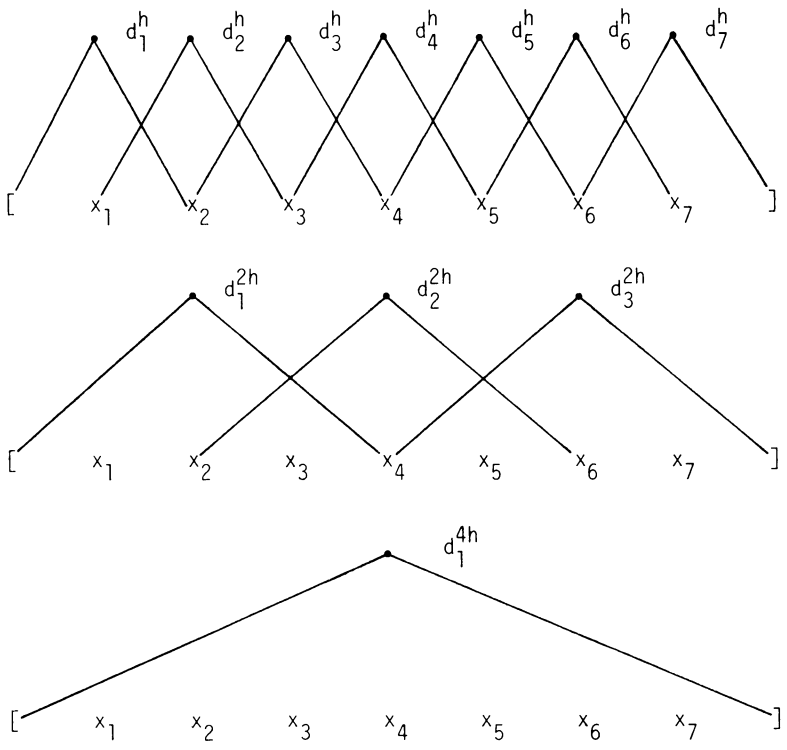

FIGURE 2

To use these directions, as with Gauss-Seidel, the objective is to subtract from u^h a proper multiple of d_k^H so that the new residual is orthogonal to d_k^H. This is given algebraically by the *directional iteration*

$$u^h \leftarrow u^h - \frac{\langle r^H, d_k^H \rangle}{\langle A^h d_k^H, d_k^H \rangle} d_k^H. \tag{2.5}$$

A simple version of unigrid can then be described in terms of one full sweep over k *and* H as follows:

For each $i = n, n-1, \ldots, 1$, let $H = 2^{-i}$ and $n^H = 2^i - 1$ and iterate via (2.5) for each $k = 1, 2, \ldots, n^H$.

Note that unigrid in this form consists of the directional iteration (2.5) with the d_k^H cycling in order through the sequence

$$\left(d_1^h, d_2^h, \ldots, d_{n^h}^h; d_1^{2h}, d_2^{2h}, \ldots, d_{n^{2h}}^{2h}; \ldots; d_1^{1/4}, d_2^{1/4}, d_3^{1/4}; d_1^{1/2}\right).$$

For a fixed "level" of these directions specified by the superscript H, this process attempts to reduce the residual error in $S_H = \text{span}\{d_k^H: 1 \leq k \leq n^H\}$. Analogous to the discussion above for $H = h$, it is effective only in terms of the transmission of information that is local *relative* to S_H. Specifically, data from a point that is further away than a few directions is slow to propagate via (2.5). (Note in Figure 2 that x_1 and x_7 are distant relative to level h but near relative to level $4h$.) Errors due to relatively long distance data connections can be quickly reduced only on the broader

levels $H' = 2H, 4H, \ldots$. (Note that S_H is a proper subset of $S_{H/2}$ so that unigrid converges here only because error components in the complement of S_H are reduced on the "narrower" levels $H' = H/2, H/4, \ldots$.)

The next section generalizes these concepts in the framework of conventional multigrid notation.

3. The General Case. As in [2], assume for focus that the problem to be solved is the d-dimensional operator equation

$$AU = f, \qquad U \in \mathcal{H}_1, \tag{3.1}$$

where A: $\mathcal{H}_1 \to \mathcal{H}_2$ is a linear operator and $\mathcal{H}_1$ and $\mathcal{H}_2$ are appropriate Hilbert spaces of functions defined on a region Ω in R^d, $d \geq 2$. Assume that (3.1) admits discretizations by a family of matrix equations, parameterized by *admissible grid sizes* $h > 0$ and given by

$$A^h U^h = f^h, \qquad U^h \in \mathcal{H}^h, \tag{3.2}$$

where $\mathcal{H}^h = R^{n^h}$ and n^h is an integer approximately proportional to h^{-d}. (Note that $n^h = h^{-1} - 1$ for the example in the previous section.) The grid transfers are full rank linear operators, represented by $I_{\bar{h}}^h$: $\mathcal{H}^{\bar{h}} \to \mathcal{H}^h$, that satisfy the consistency condition $I_{\bar{h}}^h = I_{h'}^h I_{\bar{h}}^{h'}$ for admissible $\bar{h}$, h', h when $\bar{h} < h' < h$ or $\bar{h} > h' > h$.

The class of relaxation methods considered here can be written in terms of the *directional iteration* operator G^h: $\mathcal{H}^h \times \mathcal{H}^h \to \mathcal{H}^h$ which, with relaxation parameter ω and residual $r^h = A^h u^h - f^h$, is given by

$$G^h(u^h, d^h) = u^h - \omega \frac{\langle r^h, d^h \rangle}{\langle A^h d^h, d^h \rangle} d^h. \tag{3.3}$$

The relaxation process is then specified by an ordered set of directions $\mathcal{D}^h = (\delta_1^h, \delta_2^h, \ldots, \delta_{n^h}^h) \subset \mathcal{H}^h$, so that one *sweep* with the initial guess u^h consists of iterating via

$$u^h \leftarrow G^h(u^h, d^h) \tag{3.4}$$

as d^h varies over $\mathcal{D}^h$. (Note that Gauss-Seidel is specified by the choice $\delta_k^h = e_k^h$, and with some modification this development extends to more general relaxation schemes including Jacobi.)

To define unigrid for a given admissible grid size $h = H_0$, suppose $m \geq 1$ is an integer so that $H_q = 2^q H_0$ are admissible, $q \leq m$, and define the direction sets according to

$$\mathcal{D}_q^h = \left(d_1^{H_q}, d_2^{H_q}, \ldots, d_{n^{H_q}}^{H_q} \right), \qquad 0 \leq q \leq m, \tag{3.5}$$

where $d_k^{H_q} = I_{H_q}^h d_k^{H_q}$. Thus, the directions on *level* q are just the relaxation directions on grid H_q transferred to grid $h = H_0$.

With relaxation and cycling parameters μ, ν and ν_c and present approximation u^h, then one of the many possible unigrid schemes is defined recursively by:

One unigrid cycle on level $q \geq 0$ consists first of ν unigrid relaxation sweeps via (3.4) using the ordered directions $d_k^{H_q}$, $k = 1, 2, \ldots, n^{H_q}$, followed for $q < m$ by μ cycles on level $q + 1$ and for $q = m$ by ν_c additional sweeps via (3.4).

One complete unigrid cycle applied to (3.2) on the finest level $h = H_0$ with right-hand side f^h and present approximation u^h will be denoted by $\mathrm{UG}^h(u^h, f^h)$.

Remark 1. Except possibly for determining the direction, as the name implies, unigrid operations are performed on grid $h = H_0$ only. Since the directions depend not directly on the operator A, but rather on the domain Ω, then there are usually several ways for determining them without explicit appeal to the coarser grids H_q, $q > 0$. The simplest situation in two dimensions is for a rectangular domain Ω where the discretization allows us to define I_{2h}^h as linear interpolation. With Gauss-Seidel as the fine grid relaxation, for example, and with the usual double subscript notation and $n^h = N^h \times N^h$, then $d_{k,l}^h = e_{k,l}^h$ for $h = H_0$, $1 \leqslant k, l \leqslant N^h$, and the coarse grid directions are defined recursively for $h = H_q$ by

$$d_{k,l}^{2h} = \sum_{i,j=-1}^{1} 2^{2-(|i|+|j|)} d_{2k+i,2l+j}^h. \tag{3.6}$$

Note that unigrid is invariant under a change of scale for these directions, allowing for the coefficients in (3.6) to be integers. Another of the many ways to define these directions is algebraically in terms of the i, j component of $d_{k,l}^{H_q}$ as in

$$d_{k,l}^{H_q}(i,j) = \begin{cases} (2^q - |k-i|)(2^q - |l-j|), & |k-i|, |l-j| \leqslant 2^q, \\ 0, & \text{otherwise.} \end{cases} \tag{3.7}$$

This is the approach taken in the experiments reported in Section 6.

4. Relationship to Multigrid. The unigrid cycle defined in Section 3 corresponds to the multigrid correction scheme (cf. [1]). One such multigrid cycle on problem (3.2) with present approximation u^h, right-hand side $\bar{f}^h$, and $h = H_q$ is denoted by $\mathrm{MG}_h(u^h, \bar{f}^h)$ and defined recursively by:

For $q = m$, $\mathrm{MG}_h(u^h, \bar{f}^h)$ consists of $\nu + \nu_c$ relaxation sweeps via (3.4) with directions δ_k^h, $k = 1, 2, \ldots, n^n$.

For $q < m$, $\mathrm{MG}_h(u^h, \bar{f}^h)$ consists of:

(1) Perform ν relaxation sweeps via (3.4) with directions δ_k^h, $k = 1, 2, \ldots, n^h$.

(2) Let $r^h = \bar{f}^h - A^h u^h$, $r^{2h} = I_h^{2h} r^h$, $u^{2h} \leftarrow 0$, and perform μ grid $2h$ cycles via $\mathrm{MG}_{2h}(u^{2h}, \bar{f}^{2h})$ with $\bar{f}^{2h} = r^{2h}$.

(3) Set $u^h \leftarrow u^h + I_{2h}^h u^{2h}$.

Note that $\bar{f}^h$ for $q > 1$ is the residual transferred from a finer grid approximation and u^h is an approximation to the actual error in that fine grid solution. (For $q > 1$, u^h is *not* an approximation to U^h, the solution of (3.2), since the right-hand side is f^h only when $h = H_0$.)

There are two main differences between the unigrid and multigrid techniques described. First, because only one grid is used in unigrid, its approximate solution reflects coarse level changes immediately. Thus, the corrected solution is automatically used in later computations. In the multigrid method, however, the fine grid residual is fixed at the time that the cycle progresses to coarser grids. Thus, the computations made for the coarse grid problem are not immediately used to improve the fine grid solution and update the residual. Second, to compute the low frequency corrections, unigrid uses the full solution rather than the coarse grid representation of fine grid error that MG uses.

Ignoring computational work per step, then unigrid seems preferable to multigrid since all available information is used at every stage of the method. However, in this

section it will be shown that these two approaches are in fact theoretically equivalent provided the MG formulation of the grid operators and transfers satisfies the *variational conditions*

$$A^{2h} = I_h^{2h} A^h I_{2h}^h, \tag{4.1}$$

$$I_h^{2h} = \left(I_{2h}^h\right)^T. \tag{4.2}$$

(Actually, (4.2) need only hold up to some scale factor which we omit for simplicity.)

THEOREM 1. *Under these conditions and for the same initial approximation,* u^h, *right-hand side,* f^h, *and cycling parameters,* ν, ν_c, *and* μ, *then* $\mathrm{UG}^h(u^h, f^h)$ *and* $\mathrm{MG}^h(u^h, f^h)$ *produce the same new approximation, that is, they are theoretically equivalent.*

Proof. First consider the following algorithm, intermediate to UG and MG, which for this proof is called *immediate replacement* multigrid. Using q here in place of H_q for superscripts and subscripts, then one cycle on grid H_q is denoted by $\mathrm{MGIR}_q(u^0, f^0)$ and is defined recursively by the following. (Note that MGIR is the same as MG except that each change in the coarse grid correction is immediately reflected in the fine grid approximation which is then used to update the fine grid residual and coarse grid equation.)

For $q = m$, $\mathrm{MGIR}_q(u^0, f^0)$ consists of $\nu + \nu_0$ relaxation sweeps via

$$u^0 \leftarrow u^0 + \omega \frac{\left\langle I_0^q\left(f^0 - A^0 u^0\right), \delta_k^q\right\rangle}{\left\langle A^q \delta_k^q, \delta_k^q\right\rangle} I_q^0 \delta_k^q, \qquad k = 1, 2, \ldots, n^q. \tag{4.3}$$

For $q < m$, $\mathrm{MGIR}_q(u^0, f^0)$ consists of ν relaxation sweeps via (4.3) followed by μ level $q-1$ cycles via $\mathrm{MGIR}_{q-1}(u^0, f^0)$.

Note that the immediate fine grid correction is incorporated in the relaxation scheme. This scheme on a level $q > 0$ is just (3.3) with $u^p = 0$, $0 < p \leqslant q$, and $r^q = I_0^q(f^0 - A^0 u^0)$ followed by interpolation of the correction directly to the finest grid.

To compare MG and MGIR, consider a fine grid function, $\bar{u}^0$, that monitors and incorporates each MG coarse grid computation, that is, $\bar{u}^0$ is the function that would result if MG were terminated anytime in the cycle and all coarse grid corrections were interpolated through each finer grid to properly correct the finest grid approximation u^0. (The corresponding function for MGIR is simply u^0.) In MG, assume relaxation is being performed on level $k > 0$. Let $\tilde{u}^q$ denote the current level q solution approximation for $0 \leqslant q \leqslant k$, and let $\bar{u}^q$ be the level q monitoring approximation that incorporates the corrections from coarser grids. Thus, $\bar{u}^k = \tilde{u}^k$ and $\bar{u}^q = \tilde{u}^q + I_{q+1}^q \bar{u}^{q+1}$ for $q < k$. Now suppose MG and MGIR are started with the same fine grid approximation, and consider the effect of one directional iteration during a relaxation sweep on level $k > 0$. In MGIR, the result of that iteration is

$$u_{\text{new}}^0 = u_{\text{old}}^0 + \omega \frac{\left\langle I_0^k\left(f^0 - A^0 u_{\text{old}}^0\right), d^k\right\rangle}{\left\langle A^k d^k, d^k\right\rangle} I_k^0 d^k. \tag{4.4}$$

In MG, a change of αd^k in $\tilde{u}^k$ results in a change of $\alpha I_k^0 d^k$ in $\bar{u}^0$, so the corresponding change for MG is

$$\bar{u}^0_{\text{new}} = \bar{u}^0_{\text{old}} + \omega \frac{\langle r^k, d^k \rangle}{\langle A^k d^k, d^k \rangle} I_k^0 d^k. \tag{4.5}$$

Using the recursive definitions $r^q = f^q - A^q \tilde{u}^q$ and $\bar{u}^q = \tilde{u}^q + I^q_{q+1} \bar{u}^{q+1}$, then the MG monitoring residual satisfies

$$r^k = f^k - A^k \bar{u}^k = I_0^k \left(f^0 - A^0 \bar{u}^0_{\text{old}} \right) + \sum_{q=1}^{k} I_q^k \left(I^q_{q-1} A^{q-1} I_q^{q-1} - A^q \right) \bar{u}^q.$$

By assumption (4.1), the second term in the above vanishes so that substitution into (4.5) yields

$$\bar{u}^0_{\text{new}} = \bar{u}^0_{\text{old}} + \omega \frac{\langle I_0^k \left(f^0 - A^0 \bar{u}^0_{\text{old}} \right), d^k \rangle}{\langle A^k d^k, d^k \rangle} I_k^0 d^k. \tag{4.6}$$

If both algorithms begin with the same approximation, then, by induction, (4.4) and (4.6) imply that $u^0 = \bar{u}^0$ at termination. Thus, assumption (4.1) ensures that MG and MGIR are equivalent.

To compare MGIR and unigrid, note that the unigrid directional iteration corresponding to (4.4) is given by

$$u^0_{\text{new}} = u^0_{\text{old}} + \omega \frac{\langle f^0 - A^0 u^0_{\text{old}}, I_k^0 d^k \rangle}{\langle A^0 I_k^0 d^k, I_k^0 d^k \rangle} I_k^0 d^k. \tag{4.7}$$

Assumption (4.2) then implies that the unigrid iteration is just

$$u^0_{\text{new}} = u^0_{\text{old}} + \omega \frac{\langle I_0^k \left(f^0 - A^0 u^0_{\text{old}} \right), d^k \rangle}{\langle I_0^k A^0 I_k^0 d^k, d^k \rangle} I_k^0 d^k,$$

which, with (4.1), becomes

$$u^0_{\text{new}} = u^0_{\text{old}} + \omega \frac{\langle I_0^k \left(f^0 - A^0 u^0_{\text{old}} \right), d^k \rangle}{\langle A^k d^k, d^k \rangle} I_k^0 d^k.$$

This is identical to (4.4), implying that unigrid and MGIR are equivalent methods. It follows that unigrid and multigrid are theoretically equivalent, so the proof of Theorem 1 is complete.

Although (variationally formulated) multigrid is equivalent to unigrid in that it produces the same results, there are great differences in implementation and efficiency. There are many possible implementations of unigrid with widely ranging computational characteristics, so it is difficult to make explicit comparisons, but some general differences are evident:

(1) The computational complexity of one multigrid cycle with $\mu \leqslant 3$ is $O(N)$, where N is the number of fine grid points. Unigrid cycles are significantly more expensive with complexity $O(N \log N)$ for $\mu = 1$ and $O(N^2)$ for $\mu = 2$, provided some algorithmic modifications are made.

(2) In terms of storage, unigrid essentially requires only a vector of length N, although a trade-off with efficiency can be made by providing more storage for the function f or for the directions. Usual multigrid algorithms require somewhat more

storage than this; however, the difference between unigrid and multigrid storage requirements is not generally significant.

(3) Unigrid code is typically very compact, partly because it lacks the modular structure of multigrid software. This is one reason that unigrid code can be developed very quickly. Also, there are fewer design choices with unigrid, since the coarse grid and grid transfer operators are automatically determined. This also adds to ease of programming, but restricts flexibility of the method. The design of unigrid also guarantees convergence independently of the choice of the coarse level iteration directions and cycling scheme, so mistakes may slow convergence but do not result in divergence as often as for multigrid. This may not be an asset, however.

(4) This ease of programming and small program size make unigrid an effective method to test the convergence behavior of multigrid for many application problems. It can easily be used to replace the usual relaxation or direct solvers in existing programs in order to perform such a feasibility test. Of course, the amount of work involved makes any comparison of solution time meaningless, but actual multigrid efficiency can be determined by applying the usual multigrid operation counts to the unigrid cycling scheme. Since the methods are equivalent in terms of results when multigrid is implemented according to (4.1)–(4.2), then unigrid will accurately represent the numerical performance of such a variationally formulated multigrid scheme.

(5) As with multigrid, unigrid can be extended to apply to a broader class of problems. For example, for nonlinear problems the linear relaxation in (3.4) can be replaced by the more general procedure that changes u^h by subtracting from it a proper multiple of d^h so that the new residual is orthogonal to d^h. (This will usually, but not always, require the use of an inner-loop *scalar* nonlinear solver such as Newton's method.) This extension is simpler than the nonlinear extensions of multigrid (e.g., Newton-MG and FAS) and generally more effective in accounting for the nonlinearity since all multigrid versions freeze at least some aspect of the nonlinearity during coarse grid correction.

5. Theory. For the moment, assume that A^h is symmetric and positive definite for all admissible h. Define the *energy inner-product* and *norm* on $\mathcal{H}^h$ by

$$\langle x^h, y^h \rangle_{A^h} = \langle A^h x^h, y^h \rangle$$

and

$$\|x^h\|_{A^h} = \langle A^h x^h, x^h \rangle^{1/2},$$

respectively. Let $\mathcal{W}_\lambda^h$ denote the set of all A^h-unit vectors in $\operatorname{span}\{w^h: A^h w^h = \mu w^h, \mu \leq \lambda\}$, and let $\mathcal{V}_\lambda^h$ denote its A^h-orthogonal complement. Let $\mathcal{G}^h$ signify one pass of (3.4) over the fine grid directions $\mathcal{D}_0^h$, and assume $\mathcal{D}_0^h$ spans $\mathcal{H}^h$. For each integer $\nu \geq 1$, define $\mathcal{G}_{\lambda,\nu}^h$ as the restriction of

$$(A^h)^{-1/2}\left((\mathcal{G}^h)^\nu\right)^T A^h (\mathcal{G}^h)^\nu (A^h)^{-1/2}$$

to $\mathcal{V}_\lambda^h$. (For Jacobi-type versions of (3.4), this latter operator simplifies to $(\mathcal{G}^h)^{2\nu}$.) Then, with $2m$ the degree of the differential operator in (3.1), assume (cf. [2]):

A1. There exist constants $a > 0$ and $c_0 < \infty$ independent of h such that

$$\left\|w^h - R\left(I_{2h}^h\right)\right\|_{A^h}^2 \leq c_0(\lambda h^{2m})^a$$

for all admissible h and all $w^h \in \mathfrak{W}_\lambda^h$, where $R(I_{2h}^h)$ is the range of I_{2h}^h. (Note that $R(I_{2h}^h)$ is the span of the coarse grid directions, $\mathfrak{D}_1^h$.)

A2. There exist constants c_1, $c_2 > 0$ with $c_1 < 1$ and $c_2 \leqslant (c_1 + 1)h^{-2m}/\rho(A^h)$, where $\rho(A^h)$ is the spectral radius of A^h, such that

$$\rho\left(\mathcal{G}_{\lambda,\nu}^h\right) \leqslant \max\left\{c_1^\nu, \left|1 - c_2 \lambda h^{2m}\right|^\nu\right\}.$$

THEOREM 2. *Suppose $\mu > 1$ and m and ν_c are such that the error from the coarsest level does not significantly contribute to the finest level error. Then there exists a ν independent of h such that unigrid converges to the solution of* (3.2) *by a fixed linear rate independent of h.*

Proof. This theorem follows from the results of Section 4 that relate unigrid to multigrid and from the theory of [2] slightly modified to account for the class of relaxation methods depicted in (3.4). This is a fairly straightforward proof so it is omitted.

Remark 2. A general nonsingular matrix problem can be transformed to a symmetric one via the *normal equations*

$$(A^h)^T A^h U^h = (A^h)^T f^h. \tag{5.1}$$

This is ill-advised for most methods of solution since it worsens the condition number of the problem in (3.2), but for unigrid (and multigrid) the disadvantages are usually not very great. Suppose, for example, that A^h is symmetric to begin with and that Jacobi's method with optimal relaxation parameter ω yields a smoothing rate of $\varepsilon \leqslant 0.6$, say. Then it is relatively easy to show that the smoothing rate for Jacobi with optimal parameter $\bar{\omega}$ (generally different than ω), applied to (5.1), exhibits a smoothing rate bounded by $\varepsilon^{1/4}$. This implies that four Jacobi sweeps on (5.1) reduce the oscillatory error better than one Jacobi sweep on (3.2). Since relaxation on (5.1) is significantly more expensive than on (3.2), the penalty of using (5.1) in lieu of (3.2) *for a symmetric matrix A^h* is about an order of magnitude increase in computational cost. (The smaller the smoothing rate ε, the smaller is this penalty. Note that the actual penalty depends on the details of how the unigrid directions are implemented and how the operators A^h and $(A^h)^T$ are accessed.) For matrices A^h that deviate significantly from symmetry, the extra costs are correspondingly smaller to the point that unigrid on (5.1) will converge even though on (3.2) it may not. (Note the close relationship between (5.1) and the distributed relaxation schemes in [1]. In fact, the latter corresponds to a transformation of (3.2) via

$$A^h(A^h)^T V^h = f^h, \qquad U^h = (A^h)^T V^h, \tag{5.2}$$

which has some cost advantages with multigrid. With unigrid the costs are similar. The main difference with either method is that relaxation on (5.2) minimizes actual error, and on (5.1) it minimizes residual error.)

There are several advantages in using (5.1) as opposed to (3.2) besides generality. For example, not only is convergence of symmetric schemes better understood, but error control is better; that is, it is important that relaxation on (5.1) actually minimizes the residual in the d^h direction. This is not true for (3.2) even when A^h is symmetric and positive definite, although it should be approximately so. (In fact, when direct application of unigrid to (3.2) exhibits convergence but does not

monotonically reduce the residual error on the coarse levels, this is a signal that the directions for relaxation are improperly defined. They should be chosen to approximate the *smooth* eigenvectors of A^h, that is, those that belong to the lower end of its spectrum. This would ensure that relaxation quickly eliminates the *oscillatory* eigenvector components of the error with little effect on the smooth ones. Since the spectrum of A^h that corresponds to these oscillatory components is *relatively* narrow, then there is a close relationship between error in the energy norm, for which relaxation is a minimizer, and the residual error norm. The residual norm is not generally *minimized* by relaxation, but a proper choice of directions coupled with a good smoothing rate ensures that it will be monotonically *reduced*.)

Remark 3. A major advantage in using full unigrid (or full multigrid) is that the work required on a given level is fixed so that ν need only be sufficiently large to produce the required error reduction. This is easiest to see for symmetric positive definite A^h by considering the level h truncation error in the $\mathcal{H}$-energy norm given by

$$\tau^h = \|I_h U^h - U\|_A.$$

Here, I^h: $\mathcal{H} \to \mathcal{H}^h$ and I_h: $\mathcal{H}^h \to \mathcal{H}$ are the discretization operators that, together with A, satisfy a continuous analogue of the variational conditions (4.1)–(4.2). For any $u^{2h} \in \mathcal{H}^{2h}$, with error $e^{(2h)} = I_{2h} u^{2h} - U$, then $u^h = I_{2h}^h u^{2h}$ has the same error $e^{(h)} = I_h u^h - U$, that is, $e^{(h)} = e^{(2h)}$. Thus, if u^{2h} is an approximation to U^{2h} computed so that, say,

$$\|u^{2h} - U^{2h}\|_{A^{2h}} \leqslant \tau^{2h},$$

then

$$\|e^{(2h)}\|_A \leqslant 2\tau^{2h},$$

from which it follows that

$$\|u^h - U^h\|_{A^h} = \|I_h u^h - I_h U^h\|_A \leqslant \|I_h u^h - U\|_A = \|e^{(2h)}\|_A \leqslant 2\tau^{2h}.$$

If τ^h satisfies an expression like $\tau^h \cong ch^2$ and if ν is large enough to reduce the error $\|u^h - U^h\|$ by at least a factor of 8, then full unigrid would ensure the desired result

$$\|e^{(h)}\|_A \leqslant \tau^h.$$

6. Multigrid Simulation. There are many potential uses for unigrid, perhaps as a method in its own right, but more importantly as a simulator to test the feasibility of using multigrid in a given application (especially when software already exists for solving the given problem). Attributes that favor unigrid include compact code, small storage requirements, ease of design and programming potential for vectorization and parallelism, and general guarantee of convergence. However, although it is more efficient than most methods, it is more costly than multigrid. Nevertheless, it is very useful as a multigrid simulator where it benefits from ease of implementation and automatic implementation of the variational conditions (4.1)–(4.2).

To apply unigrid to a given problem, the design question is mainly one of determining the directions in (3.5). Since a suitable choice for the interpolation operator is often dictated by the discretization process that produced the problem to be solved, then the only remaining decision is to choose the fine grid directions, that is, the relaxation process. Unigrid algorithm design is therefore a great simplification

over that for multigrid since the latter also involves careful determination of the grid transfers, coarse grid operators, and scale factors.

Some applications may require more flexibility in choosing the grid transfer and coarse grid operators than unigrid provides. However, if the discretization process that relates $\mathcal{H}^h$ to $\mathcal{H}$ is done properly, then a corresponding process can be used to relate $\mathcal{H}^{2h}$ to $\mathcal{H}^h$. That is, a proper discretization should dictate the interpolation process. For the symmetric case at least, in some sense the best choices for the coarse grid transfer and coarse grid operator are specified by the variational conditions (4.1)–(4.2). The same is true for the nonsymmetric case when unigrid is applied via the normal equations (5.1). Thus, in terms of the original problem solution, unigrid should provide a very effective design.

The inflexibility in unigrid design is precisely the same inflexibility inherent in Galerkin-type discretizations of (3.1). In fact, such discretization methods naturally satisfy (4.1)–(4.2). Unigrid automatically adopts this sort of relationship between grids regardless of how the finest grid was obtained.

There are many possibilities in the use of unigrid as a multigrid simulator. One approach is to design a portable black box that implements unigrid on a fixed region, such as a rectangle, and for a fixed relaxation method, such as Gauss-Seidel. This software developement utility could be easily *plugged* into an existing possibly very complex applications program, for example, that already uses some form of relaxation such as SOR. This would require no knowledge of the program data structure, problem origin, scaling factors or other information that multigrid implementation demands.

Another important use of unigrid as a multigrid simulator is as a research tool. The authors have developed many unigrid versions written in BASIC for the HP9845B, none of which required more than an hour of program development. Many research questions were dealt with quickly using these routines for experimentation. A few versions of these programs together with sample output are displayed in Tables 3 through 10.

The programs listed in Tables 3 and 5 apply to the problem

$$\begin{aligned} -\Delta u + g \cdot u &= f \quad \text{in } \Omega = (0, \alpha) \times (0, \beta), \\ u &= b \quad \text{on } \partial\Omega, \end{aligned} \tag{6.1}$$

where α, β are determined by the program input and the functions f, g and b are specified internally. This is similar to, but an extension of, the problem example reported in [1]. Here we use $g(x, y) = (x - y)e^{x+y-3}$, $f(x, y) = \sin(3(x + y))$, $b = \cos(3(x + y))$, and $(\alpha, \beta) = (3, 2)$.

Table 3 lists the code for the full unigrid algorithm where cycles start from the coarsest level. This process progresses to the next finer level only after a full cycle is performed on the present finest level. (It is best to use 1 as the number of cycles since full unigrid will then produce an approximation to the level of truncation error provided ν is not too small. Usually $\nu = 2$ or 3 will do. See Remark 3 of Section 5.) Note that the problem is a nine-point discretization of (6.1) and is defined in statements 480–530 and functions FNB, FNF and FNG. Modifications to these statements and functions can be easily made to apply this code to other problems. Table 4 depicts output from a sample run with full unigrid.

TABLE 3. *Full unigrid code*

```
10  DIM U(97,65),R(6)
20  DISP "# GRIDS";
30  INPUT N
40  DISP "# X POINTS, INCL. BOUNDARY POINTS";
50  INPUT I1
60  DISP "# Y POINTS, INCL. BOUNDARY POINTS";
70  INPUT J1
80  DISP "H";
90  INPUT H
100 DISP "# RELAXATIONS";
110 INPUT N0
120 DISP "CYCLING PARAMETER";
130 INPUT M0
140 DISP "# CYCLES (1 IS USUALLY BEST)";
150 INPUT C1
160 PRINT "# GRIDS=";N;" #X POINTS=";I1;" #Y POINTS=";J1;" H=";H
170 PRINT "# RELAXATIONS=";N0;" CYCLING PARAMETER=";M0;" # CYCLES";C1
180 REM ****INITIALIZE CYCLING CONTROL ARRAY
190 FOR K=1 TO N
200 R(K)=M0
210 NEXT K
220 REM ****ENFORCE BOUNDARY CONDITIONS
230 FOR I=1 TO I1
240 U(I,1)=FNB(I,1)
250 U(I,J1)=FNB(I,J1)
260 NEXT I
270 FOR J=1 TO J1
280 U(1,J)=FNB(1,J)
290 U(I1,J)=FNB(I1,J)
300 NEXT J
310 REM ****DEFINE CURRENT FINEST GRID L
320 FOR L=0 TO N-1
330 R(N-L)=C1
340 REM ****DEFINE CURRENT GRID K
350 K=N-L
360 REM ****DEFINE GRID FACTOR (M1=1 FOR FINEST GRID)
370 M1=2^(K-1)
380 REM ****FROM HERE TO STATEMENT 690, PERFORM N0 SWEEPS ON GRID K AND
390 REM          COMPUTE THE DYNAMIC RESIDUAL ERROR E
400 FOR N8=1 TO N0
410 E=0
420 FOR I=1+M1 TO I1-M1 STEP M1
430 FOR J=1+M1 TO J1-M1 STEP M1
440 A1=0
450 R1=0
460 FOR I3=I-M1+1 TO I+M1-1
470 FOR J3=J-M1+1 TO J+M1-1
480 D=8+FNG(I3,J3)*H*H*3
490 R=D*U(I3,J3)-U(I3,J3-1)-U(I3,J3+1)-U(I3-1,J3)-U(I3+1,J3)
500 R=(R-U(I3+1,J3+1)-U(I3+1,J3-1)-U(I3-1,J3+1)-U(I3-1,J3-1))/3
510 R=R-FNF(I3,J3)*H*H
520 A3=D*FND(I3,J3)-FND(I3,J3+1)-FND(I3,J3-1)-FND(I3+1,J3)-FND(I3-1,J3)
530 A3=(A3-FND(I3+1,J3+1)-FND(I3+1,J3-1)-FND(I3-1,J3+1)-FND(I3-1,J3-1))/3
540 R1=R1+FND(I3,J3)*R
550 A1=A1+FND(I3,J3)*A3
560 NEXT J3
570 NEXT I3
580 S=R1/A1
590 E=E+R1*R1
600 FOR I3=I-M1+1 TO I+M1-1
610 FOR J3=J-M1+1 TO J+M1-1
620 U(I3,J3)=U(I3,J3)-S*FND(I3,J3)
630 NEXT J3
640 NEXT I3
650 NEXT J
660 NEXT I
670 E=SQR(E)/M1/H
680 PRINT "LEVEL=";K;"  ERROR=";E
690 NEXT N8
700 REM ****THE FOLLOWING IS FOR GRID CYCLING CONTROL
710 IF K=N THEN 770
720 IF R(K)=0 THEN 760
730 R(K)=R(K)-1
740 K=K+1
750 GOTO 370
760 R(K)=M0
770 K=K-1
780 IF K>=N-L THEN 370
790 NEXT L
800 REM ****FNB DETERMINES THE BOUNDARY DATA
810 DEF FNB(I,J)=COS(3*(I+J-2)*H)
820 REM ****FND COMPUTES THE UNIGRID DIRECTIONS
830 DEF FND(I3,J3)=(M1-ABS(I-I3))*(M1-ABS(J-J3))/(M1*M1)
840 REM ****FNF IS THE RIGHT HAND SIDE OF THE EQUATION
850 DEF FNF(I3,J3)=SIN(3*(I3+J3-2)*H)
860 REM ****FNG IS THE COEFFICIENT OF THE ZERO ORDER TERM IN THE EQUATION
870 DEF FNG(I3,J3)=(I3-J3)*H*EXP((I3+J3-2)*H-3)
880  END
```

TABLE 4. *Full unigrid output*

```
#GRIDS=4
#X=25  #y=17
H=.125 #RELX=2
CYC.PARAM.=2 #CYC=1

LVL=4  ERR=1.6E0          LVL=3  ERR=9.6E-2
LVL=4  ERR=1.0E-1         LVL=3  ERR=2.2E-2
LVL=3  ERR=3.6E0          LVL=4  ERR=3.6E-3
LVL=3  ERR=5.2E-1         LVL=4  ERR=2.9E-4
LVL=4  ERR=6.2E-2         LVL=3  ERR=4.4E-3
LVL=4  ERR=5.0E-3         LVL=3  ERR=9.8E-4
LVL=3  Err=7.1E-2         LVL=4  ERR=1.5E-4
LVL=3  ERR=1.3E-2         LVL=4  ERR=6.5E-9
LVL=2  ERR=1.6E1          LVL=3  ERR=1.4E-4
LVL=2  ERR=2.7E0          LVL=3  ERR=2.4E-5
LVL=3  ERR=4.3E-1         LVL=2  ERR=1.0E-1
LVL=3  ERR=8.2E-2         LVL=2  ERR=2.5E-2
LVL=4  ERR=1.3E-2         LVL=3  ERR=4.8E-3
LVL=4  ERR=9.5E-4         LVL=3  ERR=1.1E-3
LVL=3  ERR=1.6E-2         LVL=4  ERR=1.0E-4
LVL=3  ERR=3.4E-3         LVL=4  ERR=9.4E-6
LVL=4  ERR=5.0E-4         LVL=3  ERR=2.8E-4
LVL=4  ERR=1.1E-6         LVL=3  ERR=6.2E-5
LVL=3  ERR=4.7E-4         LVL=4  ERR=9.3E-6
LVL=3  ERR=7.6E-5         LVL=4  ERR=2.4E-8
LVL=2  ERR=5.5E-1         LVL=3  ERR=1.0E-5
LVL=2  ERR=1.2E-1         LVL=3  ERR=1.7E-6
LVL=1  ERR=4.9E1          LVL=2  ERR=5.1E-3
LVL=1  ERR=1.0E1          LVL=2  ERR=1.2E-3
LVL=2  ERR=2.1E0          LVL=1  ERR=1.8E0
LVL=2  ERR=5.1E-1         LVL=1  ERR=4.0E-1
```

Table 5 contains the code for a very simple version of unigrid applied to the usual five-point discretization of (6.1). The listing shown produces the coarse-to-fine level cycling scheme depicted in column one of Table 4. Note that the problem for this sample run has 5985 unknowns. To demonstrate the effect of fine-to-coarse level cycling, statements 240 and 510 were modified by replacing $N - K$ by $K - 1$. Note the decrease in accuracy for this run. This is due to the smoothness of the initial error which for the first cycle or so favors coarse grid relaxations. Otherwise, the two cycling schemes behave almost identically as is evidenced by the convergence rates of the last few cycles.

Statements 170–210 in the sample code of Table 5 define the boundary and the initial guess for unigrid. The corresponding statements 220–300 for the full unigrid code of Table 3 define the boundary but yield zero as an initial guess (for operating systems like the HP9845B, at least). This was done to mimic the full multigrid algorithm which is usually implemented when a good guess for the grid solution is unavailable. Had this code incorporated the same statements as in Table 3, it would in effect be applying full unigrid to the fine grid residual equation

$$A^h U^h = \bar{f}^h,$$

where $\bar{f}^h = f^h - A^h b^h$ and b^h is the initial guess provided by these statements. (In fact, use of full unigrid is often advised even when b^h is a good approximation to U^h, such as for implicit discretization of time-dependent problems. This is especially true for extensions to nonlinear problems.)

TABLE 5. *Simple cycle unigrid code*

```
10 DIM U(97,65)
20 DISP "# GRIDS";
30 INPUT N
40 DISP "# X POINTS, INCL. BOUNDARY POINTS";
50 INPUT I1
60 DISP "# Y POINTS, INCL. BOUNDARY POINTS";
70 INPUT J1
80 DISP "H";
90 INPUT H
100 DISP "# RELAXATIONS";
110 INPUT N0
120 DISP "# CYCLES";
130 INPUT C1
140 PRINT "# GRIDS=";N;" #X POINTS=";I1;" #Y POINTS=";J1;" H=";H
150 PRINT "# RELAXATIONS=";N0;" # CYCLES";C1
160 C=0
170 FOR I=1 TO I1
180 FOR J=1 TO J1
190 U(I,J)=COS(3*(I+J-2)*H)
200 NEXT J
210 NEXT I
220 C=C+1
230 FOR K=1 TO N
240 M1=2^(N-K)
250 FOR N8=1 TO N0
260 E=0
270 FOR I=1+M1 TO I1-M1 STEP M1
280 FOR J=1+M1 TO J1-M1 STEP M1
290 A1=0
300 R1=0
310 FOR I3=I-M1+1 TO I+M1-1
320 FOR J3=J-M1+1 TO J+M1-1
330 D=4+EXP((I3+J3-2)*H)*H*H
340 R=D*U(I3,J3)-U(I3,J3-1)-U(I3,J3+1)-U(I3-1,J3)-U(I3+1,J3)
350 R=R-SIN(3*(I3+J3-2)*H)*H*H
360 A3=D*FND(I3,J3)-FND(I3,J3+1)-FND(I3,J3-1)-FND(I3+1,J3)-FND(I3-1,J3)
370 R1=R1+FND(I3,J3)*R
380 A1=A1+FND(I3,J3)*A3
390 NEXT J3
400 NEXT I3
410 S=R1/A1
420 E=E+R1*R1
430 FOR I3=I-M1+1 TO I+M1-1
440 FOR J3=J-M1+1 TO J+M1-1
450 U(I3,J3)=U(I3,J3)-S*FND(I3,J3)
460 NEXT J3
470 NEXT I3
480 NEXT J
490 NEXT I
500 E=SQR(E)/M1/H
510 PRINT "LEVEL=";N-K+1;"  ERROR=";E
520 NEXT N8
530 NEXT K
540 IF C<C1 THEN 220
550 DEF FND(I3,J3)=(M1-ABS(I-I3))*(M1-ABS(J-J3))/M1/M1
560  END
```

To estimate the computational complexity of a multigrid algorithm corresponding to these unigrid codes, work units (defined as in [1] as the cost of one fine grid relaxation sweep) can be accumulated for output. For Table 5, this could be done by adding statements

$$165 \quad W = 0$$
$$255 \quad W = W + (1.0/M1)$$
$$545 \quad \text{PRINT "TOTAL WORK ="; } W.$$

For the run in column one of Table 6 this would yield

$$\text{TOTAL WORK} = 8.$$

These codes determine and output the *dynamic* residual error norm using the discretized $\mathcal{L}_2(\Omega)$ norm on the grid corresponding to the level in use. However, unigrid and variationally formulated multigrid are based on minimizing the energy norm of the actual error, which is equivalent to minimizing the functional

$$F^h(u^h) = \langle A^h u^h, u^h \rangle - 2\langle f^h, u^h \rangle.$$

TABLE 6

```
#GRIDS=6 H=1/32
#X=97 #Y=65
#RELX=1 #CYC=6

LVL=6  ERR=9.2E0
LVL=5  ERR=3.7E-1
LVL=4  ERR=3.3E-1
LVL=3  ERR-3.0E-1
LVL=2  ERR=3.1E-1
LVL=1  ERR=3.5E-1
LVL=6  ERR=7.9E1
LVL=5  ERR=1.6E0
LVL=4  ERR=2.3E0
LVL=3  ERR=3.3E0
LVL=2  ERR=4.2E0
LVL=1  ERR=6.4E0
LVL=6  ERR=4.7E-2
LVL=5  ERR=9.3E-2
LVL=4  ERR=2.4E-1
LVL=3  ERR=5.7E-1
LVL=2  ERR=1.1E0
LVL=1  ERR=1.9E0
LVL=6  ERR=2.7E-3
LVL=5  ERR=8.7E-3
LVL=4  ERR=4.2E-2
LVL=3  ERR=1.2E-1
LVL=2  ERR=3.0E-1
LVL=1  ERR=6.5E-1
LVL=6  ERR=2.5E-4
LVL=5  ERR=1.4E-3
LVL=4  ERR=8.3E-3
LVL=3  ERR=2.9E-2
LVL=2  ERR=8.8E-2
LVL=1  ERR=2.4E-1
LVL=6  ERR=5.6E-5
LVL=5  ERR=3.5E-4
LVL=4  ERR=1.9E-3
LVL=3  ERR=8.1E-3
LVL=2  ERR=2.8E-2
LVL=1  ERR=9.4E-2

Simple cycle UG
output (coarse-to-
fine cycling)
```

```
#GRIDS=6 H=1/32
#X=97 #Y=65
#RELX=1 #CYC=7

LVL=1  ERR=1.4E2
LVL=2  ERR=1.2E2
LVL=3  ERR=8.4E1
LVL=4  ERR=3.6E1
LVL=5  ERR=7.0E0
LVL=6  ERR=3.2E-1
LVL=1  ERR=6.4E1
LVL=2  ERR=2.3E1
LVL=3  ERR=8.2E0
LVL=4  ERR=2.5E0
LVL=5  ERR=4.2E-1
LVL=6  ERR=1.5E0
LVL=1  ERR=2.0E1
LVL=2  ERR=5.1E0
LVL=3  ERR=1.4E0
LVL=4  ERR=2.5E-1
LVL=5  ERR=3.1E-2
LVL=6  ERR=3.4E-4
LVL=1  ERR=5.3E0
LVL=2  ERR=1.3E0
LVL=3  ERR=2.7E-1
LVL=4  ERR=3.3E-2
LVL=5  ERR=3.0E-3
LVL=6  ERR=1.9E-5
LVL=1  ERR=1.9E0
LVL=2  ERR=3.8E-1
LVL=3  ERR=5.3E-2
LVL=4  ERR=4.9E-3
LVL=5  ERR=3.4E-4
LVL=6  ERR=6.5E-6
LVL=1  ERR=7.0E-1
LVL=2  ERR=1.1E-1
LVL=3  ERR=1.2E-2
LVL=4  ERR=6.8E-4
LVL=5  ERR=4.6E-5
LVL=6  ERR=1.2E-6
LVL=1  ERR=2.7E-1
LVL=2  ERR=3.0E-2
LVL=3  ERR=3.0E-3
LVL=4  ERR=1.3E-4
LVL=5  ERR=1.2E-5
LVL=6  ERR=2.3E-7

Simple cycle UG
output (fine-to-
coarse cycling)
```

```
#GRIDS=4 H=1/8
#X=26 #Y=17 CYC.PAR.=2
#RELX=2 #CYC=1

LVL=4  ERR=1.5E0
LVL=4  ERR=6.8E-2
LVL=3  ERR=3.3E0
LVL=3  ERR=4.0E-1
LVL=4  ERR=1.6E-2
LVL=4  ERR=1.4E-3
LVL=3  ERR=5.0E-2
LVL=3  ERR=7.5E-3
LVL=2  ERR=1.4E1
LVL=2  ERR=2.2E0
LVL=3  ERR=3.4E-1
LVL=3  ERR=6.1E-2
LVL=4  ERR=3.8E-3
LVL=4  ERR=2.6E-4
LVL=3  ERR=1.6E-2
LVL=3  ERR=2.6E-4
LVL=4  ERR=3.4E-4
LVL=4  ERR=4.5E-6
LVL=3  ERR=3.6E-4
LVL=3  ERR=5.8E-5
LVL=2  ERR=4.3E-1
LVL=2  ERR=8.4E-2
LVL=1  ERR=5.1E1
LVL=1  ERR=1.3E1
LVL=2  ERR=3.4E0
LVL=2  ERR=1.1E0
LVL=3  ERR=3.4E-1
LVL=3  ERR=1.2E-1
LVL=4  ERR=2.3E-2
LVL=4  ERR=2.4E-3
LVL=3  ERR=3.0E-2
LVL=3  ERR=8.0E-3
LVL=4  ERR=1.6E-3
LVL=4  ERR=3.0E-6
LVL=3  ERR=1.6E-3
LVL=3  ERR=2.6E-4
LVL=2  ERR=2.5E-1
LVL=2  ERR=6.8E-2
LVL=3  ERR=1.4E-2
LVL=3  ERR=3.9E-3
LVL=4  ERR=5.2E-4
LVL=4  ERR=4.8E-5
LVL=3  ERR=1.2E-3
LVL=3  ERR=2.8E-4
LVL=4  ERR=4.4E-5
LVL=4  ERR=1.5E-7
LVL=3  ERR=4.8E-5
LVL=3  ERR=8.2E-6
LVL=2  ERR=1.3E-2
LVL=2  ERR=3.4E-3
LVL=1  ERR=4.8E0
LVL=1  ERR=2.3E0

Full UG output
(coarse level bounary
approximation by deflation)
```

In fact, the square of the energy norm error is just $F^h(u^h) - F^h(U^h)$. ($F^h(U^h)$ may be estimated by simple extrapolation of the $F^h(u^h)$.) It may therefore be more practical to compute this functional at the end of each relaxation sweep, expecially since it will more clearly represent the convergence behavior of the algorithm. This can be done in just a few statements and is especially useful for program debugging. (The reader is warned to be sure that the boundary conditions are incorporated in f^h.)

TABLE 7. *Full unigrid code*
(modified for coarse level boundary approximation by direction truncation)

```
10  DIM U(97,65),R(6)
20  DISP "# GRIDS";
30  INPUT N
40  DISP "# X POINTS, INCL. BOUNDARY POINTS";
50  INPUT I1
60  DISP "# Y POINTS, INCL. BOUNDARY POINTS";
70  INPUT J1
80  DISP "H";
90  INPUT H
100 DISP "# RELAXATIONS";
110 INPUT N0
120 DISP "CYCLING PARAMETER";
130 INPUT M0
140 DISP "# CYCLES (1 IS USUALLY BEST)";
150 INPUT C1
160 PRINT "# GRIDS=";N;" #X POINTS=";I1;" #Y POINTS=";J1;" H=";H
170 PRINT "# RELAXATIONS=";N0;" CYCLING PARAMETER=";M0;" # CYCLES";C1
180 REM ****INITIALIZE CYCLING CONTROL ARRAY
190 FOR K=1 TO N
200 R(K)=M0
210 NEXT K
220 REM ****ENFORCE BOUNDARY CONDITIONS
230 FOR I=1 TO I1
240 U(I,1)=FNB(I,1)
250 U(I,J1)=FNB(I,J1)
260 NEXT I
270 FOR J=1 TO J1
280 U(1,J)=FNB(1,J)
290 U(I1,J)=FNB(I1,J)
300 NEXT J
310 REM ****DEFINE CURRENT FINEST GRID L
320 FOR L=0 TO N-1
330 R(N-L)=C1
340 REM ****DEFINE CURRENT GRID K
350 K=N-L
360 REM ****DEFINE GRID FACTOR (M1=1 FOR FINEST GRID)
370 M1=2^(K-1)
380 REM ****FROM HERE TO STATEMENT 690, PERFORM N0 SWEEPS ON GRID K AND
390 REM          COMPUTE THE DYNAMIC RESIDUAL ERROR E
400 FOR N8=1 TO N0
410 E=0
420 FOR I=1+M1 TO I1-1 STEP M1
430 FOR J=1+M1 TO J1-M1 STEP M1
440 A1=0
450 R1=0
455 I2=I+M1-1
456 IF I2<I1 THEN 460
457 I2=I1-1
460 FOR I3=I-M1+1 TO I2
470 FOR J3=J-M1+1 TO J+M1-1
480 D=8+FNG(I3,J3)*H+H*3
490 R=D*U(I3,J3)-U(I3,J3-1)-U(I3,J3+1)-U(I3-1,J3)-U(I3+1,J3)
500 R=(R-U(I3+1,J3+1)-U(I3+1,J3-1)-U(I3-1,J3+1)-U(I3-1,J3-1))/3
510 R=R-FNF(I3,J3)*H*H
520 A3=D*FND(I3,J3)-FND(I3,J3+1)-FND(I3,J3-1)-FND(I3+1,J3)-FND(I3-1,J3)
530 A3=(A3-FND(I3+1,J3+1)-FND(I3+1,J3-1)-FND(I3-1,J3+1)-FND(I3-1,J3-1))/3
540 R1=R1+FND(I3,J3)*R
550 A1=A1+FND(I3,J3)*A3
560 NEXT J3
570 NEXT I3
580 S=R1/A1
590 E=E+R1*R1
600 FOR I3=I-M1+1 TO I+M1-1
610 FOR J3=J-M1+1 TO J+M1-1
620 U(I3,J3)=U(I3,J3)-S*FND(I3,J3)
630 NEXT J3
640 NEXT I3
650 NEXT J
660 NEXT I
670 E=SQR(E)/M1/H
680 PRINT "LEVEL=";K;"  ERROR=";E
690 NEXT N8
700 REM ****THE FOLLOWING IS FOR GRID CYCLING CONTROL
710 IF K=N THEN 770
720 IF R(K)=0 THEN 760
730 R(K)=R(K)-1
740 K=K+1
750 GOTO 370
760 R(K)=M0
770 K=K-1
780 IF K>=N-L THEN 370
790 NEXT L
800 REM ****FNB DETERMINES THE BOUNDARY DATA
810 DEF FNB(I,J)=COS(3*(I+J-2)*H)
820 REM ****FND COMPUTES THE UNIGRID DIRECTIONS
830 DEF FND(I3,J3)=(M1-ABS(I-I3))*(M1-ABS(J-J3))/(M1*M1)
840 REM ****FNF IS THE RIGHT HAND SIDE OF THE EQUATION
850 DEF FNF(I3,J3)=SIN(3*(I3+J3-2)*H)
860 REM ****FNG IS THE COEFFICIENT OF THE ZERO ORDER TERM IN THE EQUATION
870 DEF FNG(I3,J3)=(I3-J3)*H*EXP((I3+J3-2)*H-3)
880  END
```

TABLE 8. *Full unigrid output*
(coarse level boundary approximation by direction truncation)

```
#GRIDS=4 H=1/8
#X=26 #Y=17 CYC.PAR=2
#RELX=2 #CYC=1

LVL=4  ERR=3.5E0
LVL=4  ERR=8.8E-2
LVL=3  ERR=5.5E0
LVL=3  ERR=6.4E-1
LVL=4  ERR=4.6E-2
LVL=4  ERR=3.1E-3
LVL=3  ERR=7.2E-2
LVL=3  ERR=7.6E-3
LVL=2  ERR=1.8E1
LVL=2  ERR=2.5E0
LVL=3  ERR=3.7E-1
LVL=3  ERR=6.4E-2
LVL=4  ERR=2.4E-3
LVL=4  ERR=1.4E-4
LVL=3  ERR=1.6E-2
LVL=3  ERR=2.7E-3
LVL=4  ERR=3.8E-4
LVL=4  ERR=3.9E-6
LVL=3  ERR=4.4E-4
LVL=3  ERR=7.4E-5
LVL=2  ERR=4.5E-1
LVL=2  ERR=8.8E-2
LVL=1  ERR=4.6E1
LVL=1  ERR=7.7E0
LVL=2  ERR=1.4E0
LVL=2  ERR=3.2E-1
LVL=3  ERR=6.2E-2
LVL=3  ERR=1.3E-2
LVL=4  ERR=7.9E-5
LVL=4  ERR=1.3E-6
LVL=3  ERR=3.1E-3
LVL=3  ERR=5.6E-4
LVL=4  ERR=7.1E-5
LVL=4  ERR=5.1E-7
LVL=3  ERR=8.4E-5
LVL=3  ERR=1.5E-5
LVL=2  ERR=6.4E-2
LVL=2  ERR=1.5E-2
LVL=3  ERR=3.1E-3
LVL=3  ERR=7.6E-4
LVL=4  ERR=1.6E-5
LVL=4  ERR=1.3E-6
LVL=3  ERR=1.8E-4
LVL=3  ERR=2.7E-5
LVL=4  ERR=3.7E-6
LVL=4  ERR=4.7E-8
LVL=3  ERR=4.4E-6
LVL=3  ERR=7.1E-7
LVL=2  ERR=3.1E-3
LVL=2  ERR=6.7E-4
LVL=1  ERR=1.5E0
LVL=1  ERR=3.0E-1
```

The codes listed in Tables 3 and 5 are intended neither for efficiency nor structure. In fact, the runs depicted in Table 6 required several hours of HP9845B execution time. The main goal was ease of programming and of program modification. Dr. Bill Holland of the National Center for Atmospheric Research has a very well-developed version of unigrid that uses stencils to define the directions and is extensively modularized and structured. It is implemented on his Apple home computer and is being used to test the applicability of unigrid and multigrid to the time-dependent irregular domain diffusion problems that arise in atmospheric studies.

TABLE 9. *Nonlinear simple cycle unigrid code*

```
10 DIM U(97,65)
20 DISP "# GRIDS";
30 INPUT N
40 DISP "# X POINTS, INCL. BOUNDARY POINTS";
50 INPUT I1
60 DISP "# Y POINTS, INCL. BOUNDARY POINTS";
70 INPUT J1
80 DISP "H";
90 INPUT H
100 DISP "# RELAXATIONS";
110 INPUT N0
120 DISP "# CYCLES";
130 INPUT C1
135 DISP "LAMBDA";
136 INPUT L
140 PRINT "# GRIDS=";N;" #X POINTS=";I1;" #Y POINTS=";J1;" H=";H
150 PRINT "# RELAXATIONS=";N0;" # CYCLES=";C1;" LAMBDA=";L
160 C=0
170 FOR I=1 TO I1
180 FOR J=1 TO J1
190 U(I,J)=COS(3*(I+J-2)*H)
200 NEXT J
210 NEXT I
220 C=C+1
230 FOR K=1 TO N
240 M1=2^(N-K)
250 FOR N8=1 TO N0
260 E=0
270 FOR I=1+M1 TO I1-M1 STEP M1
280 FOR J=1+M1 TO J1-M1 STEP M1
290 A1=0
300 R1=0
310 FOR I3=I-M1+1 TO I+M1-1
320 FOR J3=J-M1+1 TO J+M1-1
330 D=4+EXP((I3+J3-2)*H)*H*H
340 R=D*U(I3,J3)-U(I3,J3-1)-U(I3,J3+1)-U(I3-1,J3)-U(I3+1,J3)
344 G=L*EXP(U(I3,J3))*H*H
345 R=R+G
350 R=R-SIN(3*(I3+J3-2)*H)*H*H
355 D=D+G
360 A3=D*FND(I3,J3)-FND(I3,J3+1)-FND(I3,J3-1)-FND(I3+1,J3)-FND(I3-1,J3)
370 R1=R1+FND(I3,J3)*R
380 A1=A1+FND(I3,J3)*A3
390 NEXT J3
400 NEXT I3
410 S=R1/A1
420 E=E+R1*R1
430 FOR I3=I-M1+1 TO I+M1-1
440 FOR J3=J-M1+1 TO J+M1-1
450 U(I3,J3)=U(I3,J3)-S*FND(I3,J3)
460 NEXT J3
470 NEXT I3
480 NEXT J
490 NEXT I
500 E=SQR(E)/M1/H
510 PRINT "LEVEL=";N-K+1;"  ERROR=";E
520 NEXT N8
530 NEXT K
540 IF C<C1 THEN 220
550 DEF FND(I3,J3)=(M1-ABS(I-I3))*(M1-ABS(J-J3))/(M1*M1)
560  END
```

7. Sample Uses. This section illustrates, by way of two examples, the ease of use of unigrid as a research tool to investigate design questions for multigrid application to a given problem.

Discretizations of nonrectangular regions often have the property that the boundary lies along lines of the grid on which the problem is to be resolved. This is a computational advantage, especially when vector computers are to be used. Unfortunately, the coarser grids needed for multigrid application do not generally share this grid alignment feature unless the coarser boundaries are subject to approximation.

TABLE 10

```
#GRIDS=3 H=1/8          #GRIDS=3 H=1/8
#X=13 #Y=9              #X=13 #Y=9
#RELX=2 #CYC=5          #RELX=2 #CYC=5
LAMBDA=.1               LAMBDA=2

LVL=3  ERR=7.3E0         LVL=3  ERR=6.7E0
LVL=3  ERR=1.2E-1        LVL=3  ERR=5.6E-2
LVL=2  ERR=1.2E1         LVL=2  ERR=1.3E1
LVL=2  ERR=2.7E0         LVL=2  ERR=2.9E0
LVL=1  ERR=1.4E1         LVL=1  ERR=1.5E1
LVL=1  ERR=3.5E0         LVL=1  ERR=3.8E0
LVL=3  ERR=4.3E-1        LVL=3  ERR=4.2E-1
LVL=3  ERR=1.9E-2        LVL=3  ERR=1.8E-2
LVL=2  ERR=9.7E-1        LVL=2  ERR=1.0E0
LVL=2  ERR=2.2E-1        LVL=2  ERR=2.2E-1
LVL=1  ERR=9.8E-1        LVL=1  ERR=1.0E0
LVL=1  ERR=2.9E-1        LVL=1  ERR=3.1E-1
LVL=3  ERR=2.6E-2        LVL=3  ERR=2.3E-2
LVL=3  ERR=2.1E-3        LVL=3  ERR=1.5E-3
LVL=2  ERR=7.8E-2        LVL=2  ERR=7.8E-2
LVL=2  ERR=1.7E-2        LVL=2  ERR=1.6E-3
LVL=1  ERR=8.3E-2        LVL=1  ERR=8.3E-2
LVL=1  ERR=2.7E-2        LVL=1  ERR=2.8E-2
LVL=3  ERR=2.3E-3        LVL=3  ERR=2.1E-3
LVL=3  ERR=1.7E-4        LVL=3  ERR=1.2E-4
LVL=2  ERR=5.9E-3        LVL=2  ERR=5.9E-3
LVL=2  ERR=1.2E-3        LVL=2  ERR=1.2E-3
LVL=1  ERR=7.6E-3        LVL=1  ERR=7.6E-3
LVL=1  ERR=2.7E-3        LVL=1  ERR=2.7E-3
LVL=3  ERR=2.1E-4        LVL=3  ERR=1.8E-4
LVL=3  ERR=9.6E-6        LVL=3  ERR=5.8E-6
LVL=2  ERR=4.8E-4        LVL=2  ERR=4.9E-4
LVL=2  ERR=7.7E-5        LVL=2  ERR=7.1E-5
LVL=1  ERR=7.5E-4        LVL=1  ERR=7.6E-4
LVL=1  ERR=2.5E-4        LVL=1  ERR=2.6E-4

Nonlinear Simple         Nonlinear Simple
Cycle Unigrid            Cycle Unigrid
Output                   Output
```

One possible grid alignment approximation scheme is to *collapse* the boundary onto grid lines and use as boundary date the (weighted) values of the finer grid solution approximation at the points on which the boundary is collapsed. To test the effect of this approach with unigrid, it is enough to specify a number of grid points in one or both directions that is not of the form $2^m + 1$, where m is the number of grids. For the sample run in column 3 of Table 6, the number of x points is 26 which means that the exact right boundary on level 2 is a half grid size from coarse grid points. Statement 420 in Table 3 therefore has the effect of ignoring those directions that extend beyond this boundary. The resulting efficiency as depicted in column 3 of Table 6 is significantly degraded over the similar run depicted in Table 4.

Another possibility is to *expand* the boundary to align it properly with the coarse grid. This is done in multigrid by defining the interpolation operator that is natural for relating these grids. For unigrid this is done simply by *clipping* the directions where they overlap the fine grid region. Table 7 illustrates a unigrid code modified for this purpose (with new statements 455–457 and changes to 420 and 460). A sample run is shown in Table 8. Note that this version of multigrid fully restores the efficiency observed in Table 4.

The second sample use of unigrid is illustrated by the code in Table 9. This program is a modification of that in Table 5 that applies to the nonlinear problem

$$-\Delta u + \lambda e^u = f \quad \text{in } \Omega = (0, \alpha) \times (0, \beta),$$
$$u = b \quad \text{on } \partial\Omega.$$

Sample runs with $\lambda = .1$ and $\lambda = 2$ are in columns 1 and 2, respectively, of Table 10.

The total *human* effort for achieving the results described in this section was less than one man-hour.

Acknowledgement. This work was supported by the National Science Foundation under grant number MCS78-03847 and the Air Force Office of Scientific Research under grant number F33615-79-C-3223.

Department of Mathematics
Colorado State University
Fort Collins, Colorado 80523

1. A. BRANDT, "Multilevel adaptive solutions to boundary value problems", *Math. Comp.*, v. 31, 1977, pp. 333–390.

2. S. F. MCCORMICK & J. W. RUGE, "Multigrid for variational problems," *SIAM J. Numer. Anal.*, v. 19, 1982, pp. 924–929.