Introduction to Multigrid Using Matlab

RANDY L. HAUPT and SUE ELLEN HAUPT

Department of Electrical Engineering, United States Air Force Academy, Colorado 80840; National Center for Atmospheric Research, Boulder, Colorado 80307

ABSTRACT

Multigrid is a powerful numerical tool for solving a variety of engineering problems. We motivate the use of the technique, introduce its theoretical basis, provide a step-by-step procedure for its use, and present examples. We include a tutorial MATLAB program, which allows the user to experiment with the technique for a typical problem. © 1994 John Wiley & Sons, Inc.

INTRODUCTION

Computational modeling of engineering problems often involves the solution of matrix equations. When the problems are very large, the solution of these equations via a direct method (Gaussian elimination) is computationally intensive and can be prone to numerical errors. Iterative methods offer an attractive alternative, because they promise to be faster, to propagate less numerical error, and to be more amenable to parallel processing. The first two promises, however, are often empty. Iterative methods are plagued with convergence problems: either they diverge or they converge quite slowly. As a result, confidence in iterative methods falls far below that of direct methods.

Multigrid is a powerful iterative method that circumvents the convergence problems of other iterative methods. As the name suggests, it uses iteration on a variety of grid sizes or matrix sizes to speed convergence. Engineers generally have little exposure to multigrid, and most of the introductory information on multigrid lies in the arcane mathematical literature. To help engineers understand multigrid, we have developed a tutorial computer program that steps the user through the full multigrid V-cycle.

Computer Applications in Engineering Education, Vol. 1(5) 421–431 (1994)
© 1994 John Wiley & Sons, Inc. CCC 1061-3773/94/050421-11

This article begins with a brief introduction to Gauss-Seidel iteration and shows why the method has convergence problems. Next, the multigrid concept is introduced, and we show why it is superior to conventional iterative methods. Finally, we introduce our graphically oriented MATLAB [1] tutorial program on multigrid. The goal is to teach students, engineers, and scientists the basics of the multigrid method.

ITERATIVE SOLUTIONS TO ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

Multigrid has applications in a wide range of problems including the solution of differential equations and integral equations [2–4]. We motivate the use of multigrid through a simple example of the iterative solution of Poisson's equation. In the x–y plane, Poisson's equation is given by

$$\nabla^2\varphi(x, y) = \vartheta(x, y) \tag{1}$$

where ∇^2 = Laplacian
φ = potential function
ϑ = sources.

Here, the problem is posed on a 2×2 square region with Dirchlet boundary conditions (potential is a constant on each side).

$$\varphi(-1, y) = c_1; \quad \varphi(1, y) = c_2; \quad \varphi(x, 1) = c_3; \quad \varphi(x, -1) = c_4 \tag{2}$$

We will investigate the iterative solution of this problem.

Assume the continuous linear elliptic partial differential equation in (1) and its associated boundary conditions are discretized. The potential is calculated at grid points spaced h apart on a uniform $N \times N$ grid superimposed on the solution region. Each grid point has an associated discrete potential given by $\phi_{mn} = \varphi(mh, nh)$, where $m, n = 1, 2, \ldots, N$. The discrete potentials are in an $N + 2 \times N + 2$ matrix with m corresponding to the matrix row and n corresponding to the matrix column. Boundary conditions occupy rows 1 and $N + 2$ and columns 1 and $N + 2$ of the matrix. The discretized Poisson's equation with the partial derivatives approximated by second-order finite difference formulas (5-point stencil) is now written as

$$\nabla_h^2 \phi(x, y) = \frac{\phi_{m+1,n} - 2\phi_{m,n} + \phi_{m-1,n}}{h^2} + \frac{\phi_{m,n-1} - 2\phi_{m,n} + \phi_{m,n-1}}{h^2} = \theta_{m,n} \tag{3}$$

where ∇_h^2 = discrete Laplacian over a grid with a spacing of h
$\phi_{m,n}$ = potential at row m and column n
$\theta_{m,n}$ = source term
h = spacing between grid points in the x and y directions

This equation may be put in matrix form and solved directly using a Gaussian elimination algorithm. Gaussian elimination takes on the order of N^3 flops (floating point operations) to arrive at a solution.

Alternatively, the equation may be solved iteratively. Each iteration step takes on the order of N^2 flops. If the solution converges in less than on the order of N steps, then the iterative method beats Gaussian elimination. This speed potential makes iterative methods very attractive. Equation (3) can be rewritten as

$$\phi_{m,n} = \tfrac{1}{4}(\phi_{m+1,n} + \phi_{m-1,n} + \phi_{m,n+1} + \phi_{m,n-1} - h^2\theta_{m,n}) \tag{4}$$

The $\phi_{m,n}$ and $\theta_{m,n}$ are the elements of matrices Φ and Θ, respectively, where the subscripts m, n denote the index of the x and y coordinates, respectively. One way to solve this equation iteratively is to calculate a new value, $\phi^i_{m,n}$, from the four surrounding old values, $\phi^{i-1}_{m+1,n}$, $\phi^{i-1}_{m-1,n}$, $\phi^{i-1}_{m,n+1}$, and $\phi^{i-1}_{m,n-1}$ (superscripts give the iteration number). The method is known as Jacobi iteration [6], and (4) is written as

$$\phi^i_{m,n} = \tfrac{1}{4}(\phi^{i-1}_{m+1,n} + \phi^{i-1}_{m-1,n} + \phi^{i-1}_{m,n+1} + \phi^{i-1}_{m,n-1} - h^2\theta_{m,n}) \tag{5}$$

An algorithm begins with an initial guess, Φ^0. (Often a first guess of all zeros is as good a starting point as any other.) Each iteration calculates new matrix elements for Φ^i at all the grid points. This method requires storing two matrices: Φ^i and Φ^{i-1}. A logical improvement to the Jacobi iteration updates $\phi^{i-1}_{m,n}$ is generated, thus only requiring one matrix to store the discrete values of $\phi_{m,n}$. This method is known as Gauss-Sidel [6] iteration, and is written as

$$\phi^i_{m,n} = \tfrac{1}{4}(\phi^{i-1}_{m+1,n} + \phi^{i}_{m-1,n} + \phi^{i-1}_{m,n+1} + \phi^{i}_{m,n-1} - h^2\theta_{m,n}) \tag{6}$$

Gauss-Seidel generally produces quicker convergence than the Jacobi method. A common variation to the above iteration is known as red–black Gauss-Seidel iteration [7]. This method breaks the grid into two superimposed grids like a checker board as shown below:

$$\begin{array}{ccccc}
\mathbf{R} & \boxed{B} & \mathbf{R} & B & \mathbf{R} \\
\boxed{B} & \boxed{\mathbf{R}} & \boxed{B} & \mathbf{R} & B \\
\mathbf{R} & \boxed{B} & \mathbf{R} & B & \mathbf{R} \\
B & \mathbf{R} & B & \mathbf{R} & B \\
\mathbf{R} & B & \mathbf{R} & B & \mathbf{R}
\end{array}$$

The black grid (B) consists of points that are updated using only the points from the red grid (**R**). Likewise, the red grid points are updated using only the black grid points. For example, the boxed red grid point shown is calculated from the surrounding four boxed black grid points. The primary advantage of this method is that it is easy to implement on a parallel computer.

As an example, consider the problem of finding the potential inside a square region (4 m^2) having three of its sides ($x = -1$, $x = 1$, and $y = -1$) at 0 and the side at $y = 1$ at 5. This problem could represent a square region containing a charge density, where the electric potential is at 5 V on one side, and the other three sides are grounded. It could also represent fluid potential flow in a basin with three closed sides, and the remaining side open to a flow with constant potential. The eigenfunction expansion solution of this problem is given by

$$\varphi(x, y) = \frac{20}{\pi} \sum_{i=\text{odd}}^{\infty} \frac{\sin(i\pi x)\sinh(i\pi y)}{i \sinh(i\pi)} \quad (7)$$

This expression serves as the "exact solution" with which to compare the results of the numerical methods.

A 17 × 17 grid is superimposed over the solution region to apply Gauss-Seidel iteration. Figures 1a–c show contour plots of the potential on the grid after 10, 50, and 100 iterations, respectively. The normalized l_2 norm of the error between the Gauss-Seidel iteration and the eigenfunction expansion is given by

$$l_2 = \frac{\sqrt{\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} |\varphi(nh, mh) - \phi^i_{m,n}|^2}}{\sqrt{\sum_{n=0}^{\infty} \sum_{m=0}^{\infty} |\varphi(nh, mh)|^2}} \quad (8)$$

This error reduces from 0.3668 to 0.0706 to 0.0106 in Figures 1a–c, respectively. The solid line in Figure 2 is the l_2 norm of the error for various numbers of iterations. Convergence is very fast for the first few iterations; then convergence becomes very slow for the remaining iterations.

THE PROBLEM WITH GAUSS-SEIDEL ITERATION: MOTIVATION FOR MULTIGRID

The previous example of an iterative solution converged slowly and did not outperform Gaussian

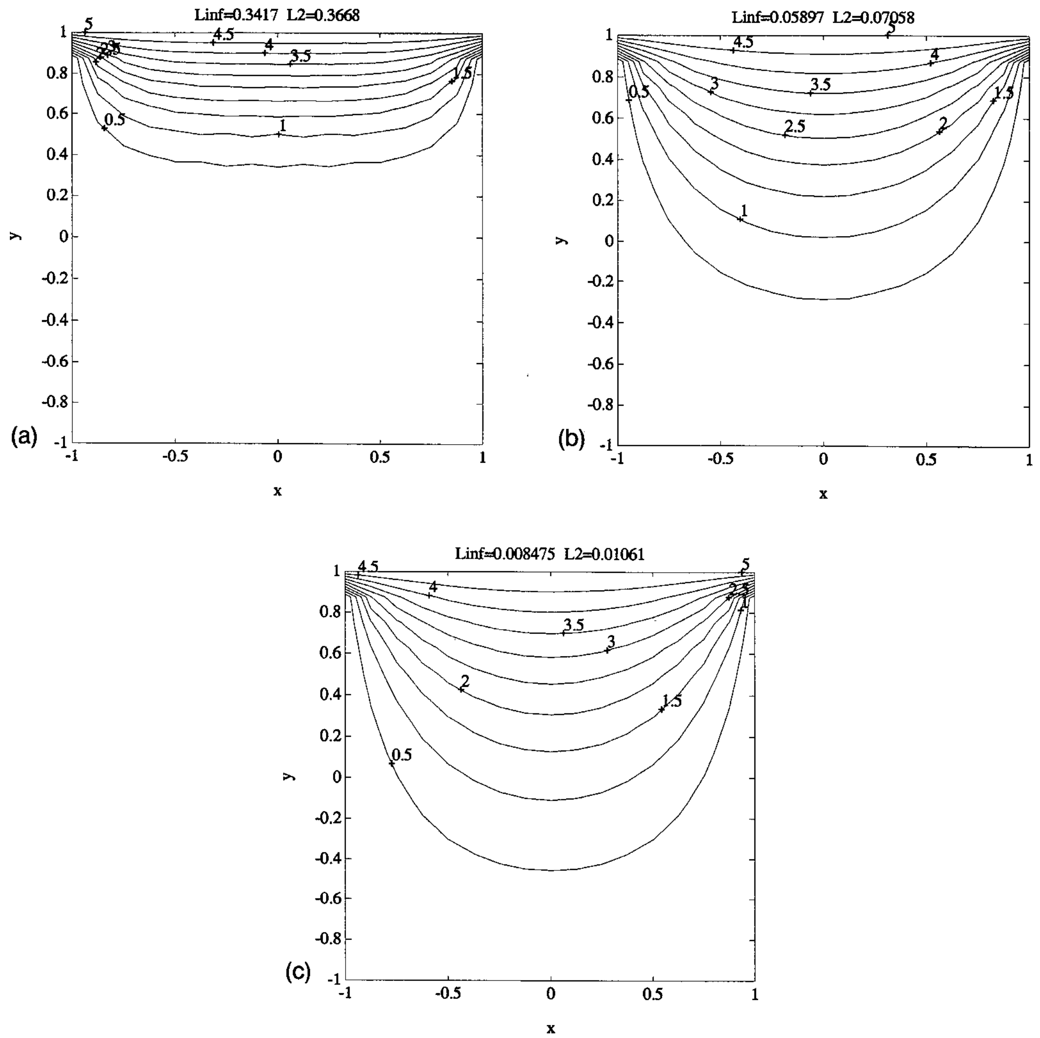


Figure 1 Contour plots of the solution after (a) 10, (b) 50, and (c) 100 Gauss-Seidel iterations on a 17 × 17 grid. The Dirchlet boundary conditions are 1 on the top side and 0 on the other three sides.

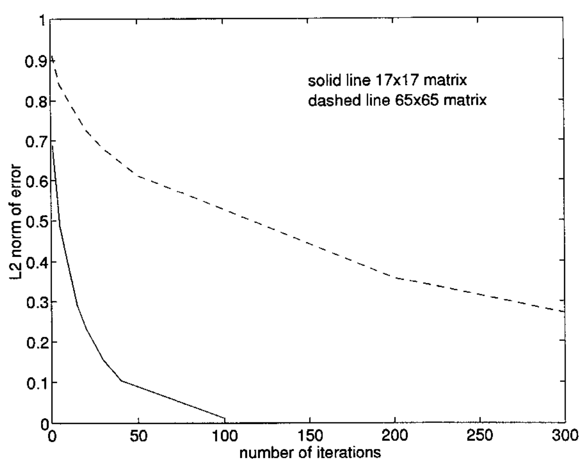


Figure 2 Graph of the l_2 norm versus number of Gauss-Seidel iterations for the problem on a 17×17 grid (solid line) and for the problem on a 65×65 grid (dashed line).

elimination. Using a finer grid produces a larger matrix that takes many more iterations to converge to the same error level. This section explains the slow convergence problem of iterative methods, thus setting the stage for multigrid.

Iterative methods start with an initial guess for $\phi_{m,n}$ represented by $\phi^0_{m,n}$. After each iteration, $\phi^i_{m,n}$ is updated to a closer approximation to the exact solution, $\varphi(mh, nh)$. The error at step i in the iteration is given by

$$\epsilon^i_{m,n} = \varphi(nh, mh) - \phi^i_{m,n} \tag{9}$$

Iterative methods dramatically reduce the error in the first few steps, then take many additional steps to reduce the remaining error. The slow convergence problem of Gauss-Seidel iteration can be explained by examining the error in the frequency domain [7]. Expanding the error in a Fourier half sine series gives

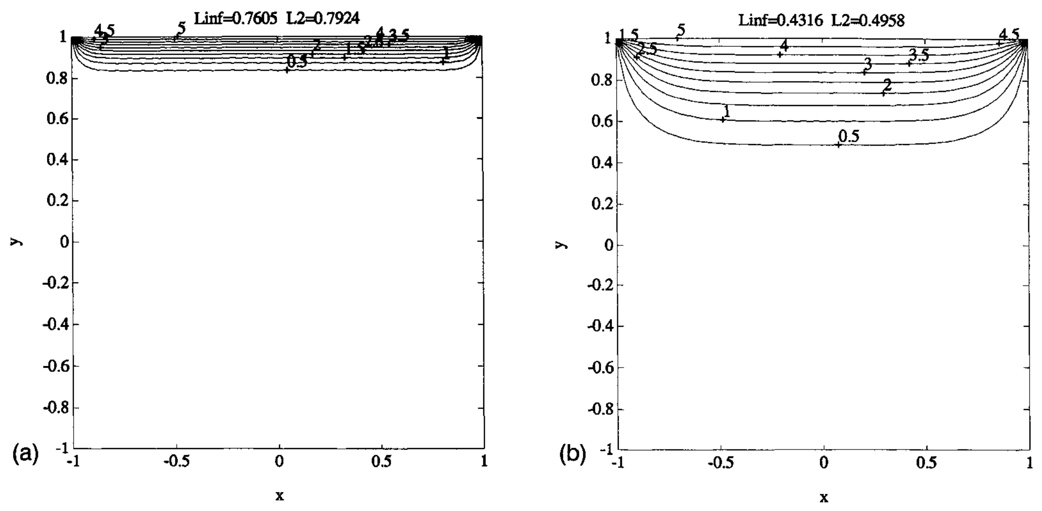
$$\epsilon_{m,n} = \sum_{k_x=1}^{N-1} \sum_{k_y=1}^{N-1} \sin\left(\frac{mk_x\pi}{N}\right)\sin\left(\frac{nk_y\pi}{N}\right) \tag{10}$$
$$0 \le m, n \le N$$

Each $\epsilon_{m,n}$ corresponds to a point on the $N \times N$ finite difference grid. The integers k_x and k_y are wavenumbers that indicate the number of half sine waves across the grid in the x and y directions, respectively. When k_x or k_y is small the error terms are low frequency or smooth, and when k_x or k_y is large the error terms are high frequency or oscillatory. Because a smooth error term has a smaller change in value between adjacent grid points, relaxation techniques calculate a smaller correction term (the finite difference is very small) for the smooth mode, and convergence is slow. An oscillatory mode has a larger difference between the error term values at adjacent grid points, and the relaxation technique calculates larger correction terms that result in faster convergence. In Figure 2 (solid line) the large reduction in the l_2 norm in the first few iterations is due to the quick damping of the oscillatory error terms. The dramatic slow down in convergence is due to the slow damping of the smooth error terms. The highest modes may also slow the convergence of the iterative algorithm because they are aliased. Aliasing occurs when high frequency modes are misinterpreted as low frequency modes due to inadequate sampling.

Consider the previous example as the number of grid points increases. A 65×65 grid is superimposed over the solution region and Gauss-Seidel iteration applied. Figures 3a–c show contour plots of the potential on the grid after 10, 100, and 500 iterations, respectively. The l_2 norm of the error between the Gauss-Seidel iteration and the eigenfunction expansion solution varies from .7924 to .4958 to .1628 in Figures 3a–c, respectively. Convergence is slow for this larger grid size. the dashed line of Figure 2 is a graph of the l_2 norm of the error as a function of iterations. Even when normalized by the number of grid points, the error is much worse for the larger grid. Figure 4 is a graph of the l_2 norm of the error versus the number of iterations divided by matrix size for three matrix sizes. Even when the number of iterations is normalized by the size of the matrix, the larger matrices take considerably longer to converge. Therefore, attempts to increase resolution of a problem solved by iterative methods are often confounded by slowing of convergence and an increase in the error. This paradox is a direct result of adding more high frequency components, making it more difficult to smooth the slow components.

MULTIGRID SOLUTIONS TO ELLIPTIC PARTIAL DIFFERENTIAL EQUATIONS

The previous example vividly shows how convergence of the Gauss-Seidel method slows as the grid size increases. Multigrid prevents this slowing by iterating on grids of various sizes; thus, it substantially reduces all frequency components of the error. For instance, if a square solution space is superimposed with grids that have spacings of h, $2h$, $4h$, and $8h$ (see Fig. 5), then multigrid uses Gauss-Seidel iteration at each of these grids to reduce a wide range of frequencies in the error term. In Figure 5, four grids are superimposed on the square solution region



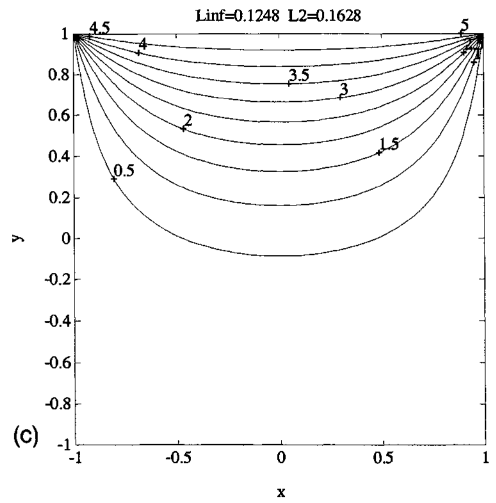


Figure 3 Contour plots of the solution after (a) 10, (b) 100, and (c) 500 Gauss-Seidel iterations on a 65 × 65 grid. The Dirchlet boundary conditions are 1 on the top side and 0 on the other three sides.

in the x–y plane. The first grid, $G(8h)$, is the coarsest grid with a spacing of $8h$ between grid points. After iterating on the coarsest grid just a few times, the approximate solution is interpolated to the next finest grid, $G(4h)$. Again, a few Gauss-Seidel iterations are applied. Instead of just interpolating to the next finest grid, multigrid returns to the coarsest grid to correct the solution at $G(8h)$. Next, the correction is interpolated to $G(4h)$ and Gauss-Seidel iteration is applied. One V-cycle has been completed and the solution at $G(4h)$ has been found. If a solution at a finer grid spacing is necessary, then the $G(4h)$ solution is interpolated to $G(2h)$ and another V-cycle is accomplished. Repeating V-cycles in this manner to get the solution at the finest grid, $G(h)$, is known as the full multigrid V-cycle (FMV-cycle). The strategy of multigrid is to perform only a few iterations at the fine grid, because that grid takes the most computation time. By performing most of the iterations at the coarser grids, the number of flops to achieve convergence can be substantially reduced.

Now we present the details of the multigrid algorithm. The exact solution to the discrete Poisson equation at grid h is Φ_h^*, where

$$\nabla_h^2 \Phi_h^* = \Theta_h \qquad (11)$$

The residual after iteration i is given by

$$R_h^i = \Theta_h - \nabla_h^2 \Phi_h^i \qquad (12)$$

The error is then

$$E_h^i = \Phi_h^* - \Phi_h^i \qquad (13)$$

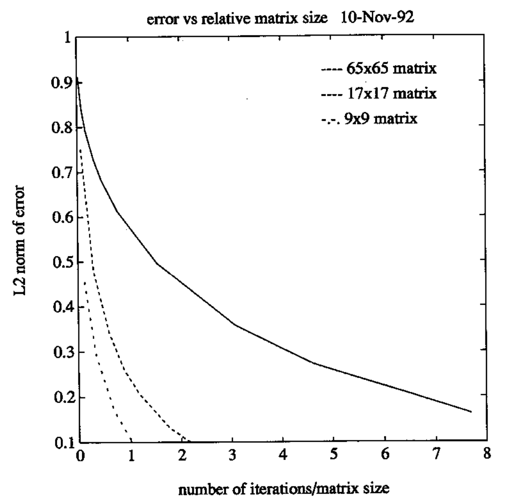


Figure 4 Graph of the l_2 norm versus number of Gauss-Seidel iterations for the problem in Figure 3.

Combining equations (11), (12), and (13) gives the error equation

$$\nabla^2 E_h^i = R_h^i \tag{14}$$

Therefore, once initial iterations have been performed to compute the residual, one may directly compute the error by relaxation on equation (14). The error is then added to Φ_h^i to estimate Φ^* by equation (13). The solution of the error equation is itself solved by an embedded iteration, as detailed subsequently.

We will now show the details in a step-by-step process. We will define $G(h)$ to have $2^n + 1 \times 2^n + 1$ points, where the grid has n increments h wide in both directions. The finest grid is $G(h)$ and the coarsest grid is $G(h2^{lo})$, where $2^{lo-1} \geq 1$ and lo is the index of the coarsest grid. A full multigrid V-cycle begins at $G(h2^{lo})$ and continues until a final solution at $G(h)$ is obtained.

Figure 6 shows the flow of the FMV algorithm from a $4h \times 4h$ grid to an $h \times h$ grid. It will be useful to refer to this figure as you read the FMV algorithm steps below, because it includes the equations used at each step.

Step 1.

Starting at the coarsest grid, make an initial guess to the solution (Φ_{4h}^0). Usually, $\Phi_{4h}^0 = 0$ works well. Iterate i_0 times with equation (6) to arrive at an approximate solution on this grid, Φ_{4h}. This is the best solution available at this grid level, so it is double-boxed in Figure 6. Next, interpolate the solution (Φ_{4h}) and sources (θ_{4h}) from $G(4h)$ to $G(2h)$. The operator I_{4h}^{2h} interpolates a matrix on the $4h \times 4h$ grid to a matrix on the $2h \times 2h$ grid. Usually, linear interpolation is used, but some higher order interpolation, such as bicubic splines, can also be used.

Step 2

Apply i_1 Gauss-Seidel iterations on $G(2h)$. Calculate the residual on $G(2h)$ using equation (12). Next, restrict the residual from $G(2h)$ to $G(4h)$. Restriction is a means of condensing a larger matrix into a smaller matrix. Two common brands of restriction are injection and full weighting. Injection takes every other value from the larger matrix and transfers it to the smaller matrix. Full weighting takes a weighted average of every other value and its neighbors from the larger matrix and transfers it to the smaller matrix. The full weighted restriction of an element in the residual matrix is given by [7]:

$$\begin{aligned} I_{2h}^{4h} r_{4m+1,4n+1} = \tfrac{1}{16}\{&4r_{2m+1,2n+1} \\ &+ 2[r_{6m+1,4n+1} + r_{4m+1,6n+1} + r_{2m+1,4n+1} \\ &+ r_{4m+1,2n+1}] + r_{6m+1,2n+1} + r_{6m+1,6n+1} \\ &+ r_{2m+1,2n+1} + r_{2m+1,6n+1}\} \end{aligned} \tag{15}$$

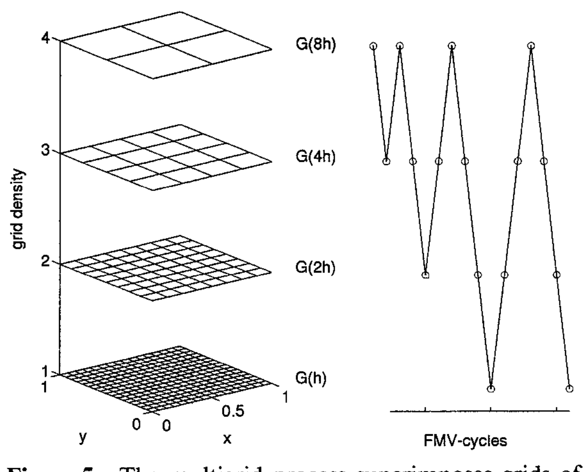


Figure 5 The multigrid process superimposes grids of various sizes over the solution region. On the left are four grids with spacings between evaluation points of h, $2h$, $4h$, and $8h$. Multigrid solves the problem on each grid level and uses interpolation and restriction to move information about the problem among grids. To the right of the grids is a simple diagram of the FMV-cycle. A problem is first solved on the coarse grid ($G[8h]$) and the final solution is on the fine grid ($G[h]$). The algorithm follows the V-cycle path between coarse and fine grids from $G(8h)$ to $G(h)$.

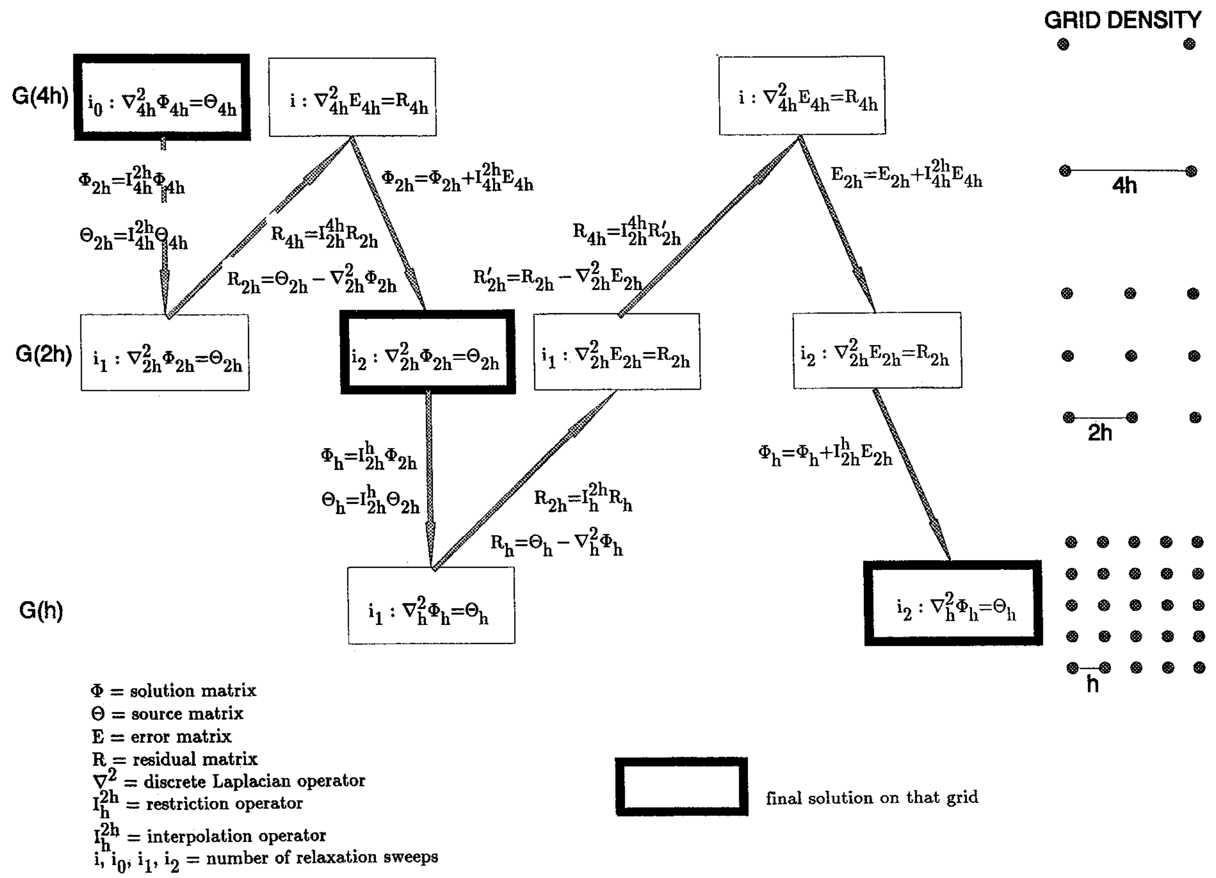

GRID DENSITY
G(4h)
G(2h)
G(h)
$i_0 : \nabla^2_{4h}\Phi_{4h}=\Theta_{4h}$
$i : \nabla^2_{4h}E_{4h}=R_{4h}$
$i : \nabla^2_{4h}E_{4h}=R_{4h}$
$\Phi_{2h}=I^{2h}_{4h}\Phi_{4h}$
$\Theta_{2h}=I^{2h}_{4h}\Theta_{4h}$
$\Phi_{2h}=\Phi_{2h}+I^{2h}_{4h}E_{4h}$
$R_{4h}=I^{4h}_{2h}R_{2h}$
$R_{2h}=\Theta_{2h}-\nabla^2_{2h}\Phi_{2h}$
$R_{4h}=I^{4h}_{2h}R'_{2h}$
$R'_{2h}=R_{2h}-\nabla^2_{2h}E_{2h}$
$E_{2h}=E_{2h}+I^{2h}_{4h}E_{4h}$
$i_1 : \nabla^2_{2h}\Phi_{2h}=\Theta_{2h}$
$i_2 : \nabla^2_{2h}\Phi_{2h}=\Theta_{2h}$
$i_1 : \nabla^2_{2h}E_{2h}=R_{2h}$
$i_2 : \nabla^2_{2h}E_{2h}=R_{2h}$
$\Phi_h=I^h_{2h}\Phi_{2h}$
$\Theta_h=I^h_{2h}\Theta_{2h}$
$R_{2h}=I^{2h}_h R_h$
$R_h=\Theta_h-\nabla^2_h\Phi_h$
$\Phi_h=\Phi_h+I^h_{2h}E_{2h}$
$i_1 : \nabla^2_h\Phi_h=\Theta_h$
$i_2 : \nabla^2_h\Phi_h=\Theta_h$
4h
2h
h
Φ = solution matrix
Θ = source matrix
E = error matrix
R = residual matrix
∇^2 = discrete Laplacian operator
I^{2h}_h = restriction operator
I^{2h}_h = interpolation operator
i, i_0, i_1, i_2 = number of relaxation sweeps
final solution on that grid

Figure 6 A diagram of one possible configuration of an FMV-cycle. Grid density increases from top to bottom. The solution on the coarsest grid is in the upper-left double box and the final solution is in the lower-right double box. The double box indicates that the solution is the best that is obtained at that grid level. Moving up in the diagram requires restriction, whereas moving down requires interpolation. The subscript on the restriction and interpolation operators is the current grid size, the superscript is the next grid size. The superscript notation on all vectors and matrices is the current grid size.

The increased convergence speed of full weighting usually offsets its increased operation count.

Step 3.

Perform $i = i_1 + i_2$ Gauss-Seidel iterations on the error equation (14) at $G(4h)$. Interpolate the error from $G(4h)$ to $G(2h)$ and add it to the approximate solution (13) at $G(2h)$ found in step 2. Adding this interpolated version of the error to the solution at $G(2h)$ corrects the low-frequency error components of Φ_{2h}.

Step 4

Iterate i_2 times to obtain the final solution at $G(2h)$ and interpolate the solution and sources to $G(h)$.

Step 5

Apply i_1 Gauss-Seidel iterations on $G(h)$. Calculate the residual on $G(h)$. Next, restrict the residual from $G(h)$ to $G(2h)$.

Step 6

Perform i_1 Gauss-Seidel iterations on the error equation at $G(2h)$. Calculate the residual on $G(2h)$. Next, restrict the residual from $G(2h)$ to $G(4h)$.

Step 7

Perform i Gauss-Seidel iterations on the error equation at $G(4h)$. Interpolate the error from $G(4h)$ to $G(2h)$ and add to the approximate error calculated at $G(2h)$. Adding this interpolated version of the error corrects the low-frequency error components to E_{2h}.

Step 8

Perform i_2 Gauss-Seidel iterations on the error equation at $G(2h)$. Interpolate the error from $G(2h)$ to $G(h)$ and add to the approximate solution at $G(h)$ found in step 5. Adding this interpolated version of the error corrects the low-frequency error components of Φ_h.

Step 9

Iterate i_2 times to obtain the final solution at $G(h)$.

This process is repetitive and ideally suited for recursive subroutines.

FMV MATLAB PROGRAM

The MATLAB software uses the FMV cycle shown in Figure 6. The program begins by asking for the following information:

- grid size
  - lowest grid level, $lo \geq 1$, number of grid points = $[2^{lo} + 1]^2$
  - highest grid level, hi, number of grid points = $[2^{hi} + 1]^2$
- restriction
  - full-weighting
  - injection
- number of relaxation steps
  - initially, i_0
  - on the downsweep, i_1
  - on the upsweep, i_2
- Position and strength of source points

The following m-files are part of the FMV algorithm:

- FMV.m—main program; performs data input and output
- restrict.m—performs injection or full-weighting restriction
- interp.m—performs bilinear interpolation
- relax.m—performs red–black Gauss-Seidel relaxation
- vcycle.m—performs one complete *V*-cycle
- resid.m—calculates the residual
- dirchlet.m—enter the desired dirchlet boundary conditions in this program
- lap2d.m—calculates the eigenfunction expansion solution

Because MATLAB allows recursive function calls, the matrices can have different dimensions at each grid level; thus, no memory space is wasted. (A FORTRAN 77 program does not have this luxury of recursive calls. The usual method is to initialize all the Φ and Θ values at once in one vector and to pass this vector plus a pointer for the location of the current grid to the various subroutines. If a FORTRAN 77 program used Φ and Θ matrices, then these matrices would all have the same dimension at each grid level, even though only a portion of them were filled.) A disadvantage to formulating Φ and Θ as matrices is that the speed of the solution cannot be readily compared to Gaussian elimination (in which Φ and Θ are formulated as vectors), and nonrectangular regions are inefficiently represented.

If the multigrid solution has *nlev* levels (or number of different sized grids), then there are *nlev*-1 *V*-cycles in one FMV cycle. The FMV cycle consists of an *nlev*-1 loop that cells a *V*-cycle function from within. Each step in the loop has a larger *V*-cycle (extending over more levels).

Output plays an important role in this tutorial algorithm. The following values are printed while the MATLAB program is running:

- contour plot of boundary values
- Φ, E, and R at each block in Figure 7
- contour plots of Φ, E, and R at the end of each *V*-cycle
- plots of the norms of the error

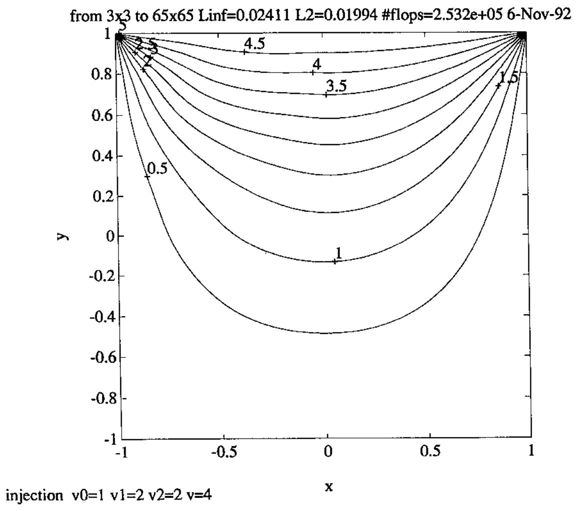
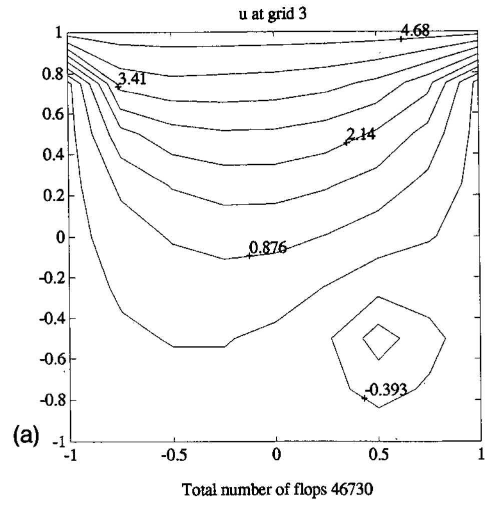


Figure 7 Multigrid solution to the Laplace equation problem in Figure 3.



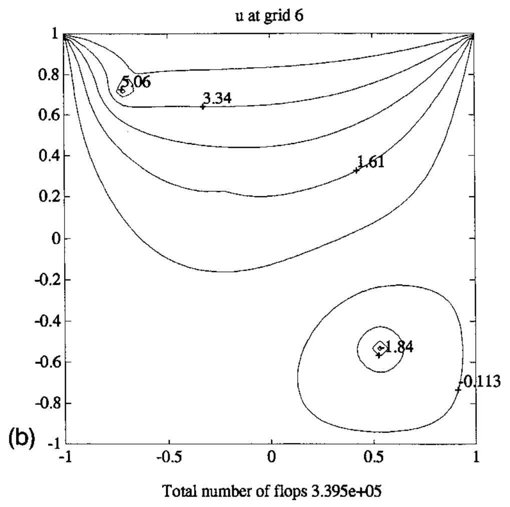


Figure 8 Multigrid solution at grid levels (a) 3 and (b) 6.

The l_2 and l_∞ error norms in the algorithm are defined as

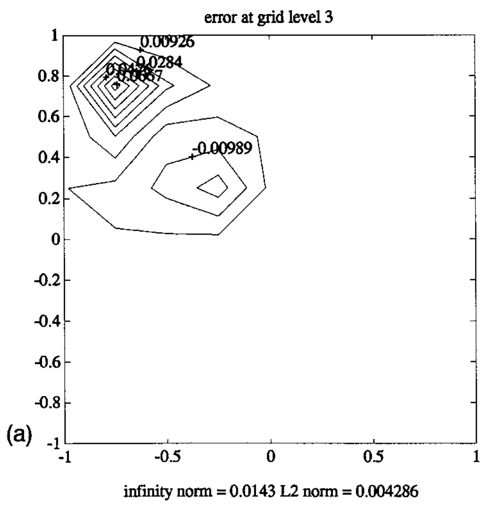
$$l_2 = \frac{\sqrt{\sum_{m=1}^{N}\sum_{n=1}^{N} e_{m,n}^2}}{\sqrt{\sum_{m=1}^{N}\sum_{n=1}^{N} \phi_{m,n}^2}} \qquad l_\infty = \frac{\max|e_{m,n}|}{\max|\phi_{m,n}|}$$

where N = number of matrix elements
$e_{m,n}$ = error associated with matrix element n
$\phi_{m,n}$ = solution associated with matrix element n.

ese error norms differ from that in equation (8), because we no longer have an "exact" solution for comparison.

To demonstrate the MATLAB program in action, we solved the Laplace equation example using multigrid. Figure 7 shows the converged results for a 65 × 65 grid after 253,000 flops. The FMV-cycle started with a 3 × 3 matrix and $i_0 = 5$, $i_1 = 2$, and $i_2 = 2$. Multigrid required less than N^3 flops to solve this problem accurately.

The next example has one source and one sink point added. The Dirchlet boundary condition at $y = 1$ is 5, whereas all the other boundaries are at 0. Figure 8a shows the potential at grid level 3 and Figure 8b shows the potential at grid level 6. The MATLAB program displays a contour plot of the error and potential at the final step in each V-cycle. Figures 9a and b are the contour plots of the error corresponding to the potential contours in Figure 8a and b. Figure 10 shows the extremely fast convergence of the multigrid algorithm for this problem, even for the difficult resolution of 65 × 65 for grid 6. Even for the small number of iterations per grid,



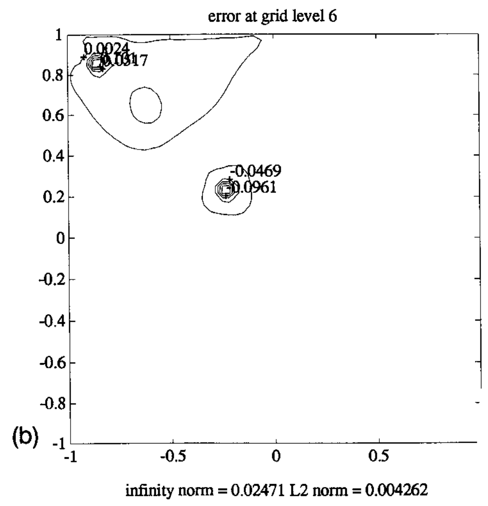


Figure 9 Contour plots of the error at grid levels (a) 3 and (b) 6.

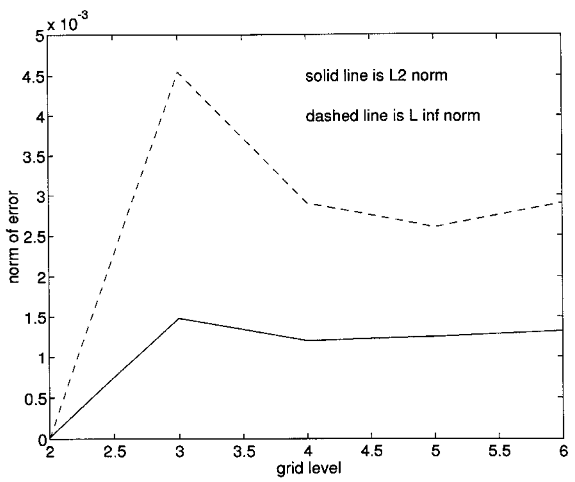


Figure 10 Plot of the l_2 (solid line) and $l\infty$ (dashed) norms of the error.

convergence on each is on the order of 10^{-3}. This is because the fast components are well-smoothed on the coarse grid before proceeding to finer grids.

This routine allows easy sensitivity analysis to different ways of doing the problems. Some lessons learned include: 1) always start with the coarsest possible grid (3×3) and 2) use full-weighting. These options add little computational expense for a better solution. The value of multigrid over Gauss-Seidel and Gaussian elimination becomes more obvious as the matrix size increases.

SUMMARY

Multigrid is an efficient tool for solving many engineering problems. Unfortunately, it is not yet widely applied because of the lack of wide availability of "black box" routines and the difficulty of applying the technique for the first time. Even the few black box routines that do exist require a cursory understanding of multigrid for proper application.

The goal of the software developed here is to be used as a hands-on training tool for scientists and engineers who want to understand the basics of multigrid quickly. This article serves as a user's manual for the software package, which will be included in a diskette to be distributed in a future issue of *CAE*, as well as a brief motivation and introduction to the concepts behind multigrid.

It should be noted that multigrid is by no means limited to the simple Poisson equation in (1). It has been applied to widely varying problems in a variety of fields. It can be used for complicated systems of equations, time varying problems, integral equations, non-elliptic systems, adaptive grids, and nonlinear problems. It is often applied with other discretization techniques and nonuniform spacing. The even wider concept of multilevel techniques uses the general concept of many grids to solve a problem more efficiently. These advanced applications are beyond the scope of this article. Our hope is that we have provided an accessible demonstration of multigrid that will motivate the readers to find applications in their own problems.

ACKNOWLEDGMENTS

S. E. Haupt was partially funded by the National Science Foundation under Grant ATM-90-11413. The National Center for Atmospheric Research is funded by the National Science Foundation.

REFERENCES

[1] *MATLAB Reference Guide.* The MathWorks, Inc., Natick, MA, 1992.

[2] S. McCormick, *Multigrid Methods.* Vol. 3 of SIAM Frontiers Series, SIAM, Philadelphia.

[3] S. McCormick, *Multilevel Adaptive Methods for Partial Differential Equations.* SIAM, Philadelphia, 1989.

[4] A Brandt, *Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics.* GMD, Postfach 1240, D-5205, St. Augustin, 1, F.R., Germany, 1984.

[5] K. Kalbasi, "A DSP-Based Multilevel Iterative Technique for the Moment Method Solution of Large Electromagnetic Problems," PhD Dissertation, The University of Kansas, 1992.

[6] W. Press, et al., *Numerical Recipes, 2nd. Ed,* Cambridge University Press, New York, 1992.

[7] W. Briggs, *A Multigrid Tutorial,* SIAM, Philadelphia, 1987.

BIOGRAPHIES

Randy L. Haupt received the B.S. degree in Electrical Engineering from the United States Air Force Academy, Colorado, the M.S. degree in Engineering Management from Western New England College, Springfield, Massachusetts, the M.S. degree in Electrical Engineering from Northeastern University, Boston, Massachusetts, and the Ph.D. Degree in Electrical Engineering from The University of Michigan, Ann Arbor, in 1978, 1981, 1983, and 1987, respectively.

From 1978–1980, he worked as an engineer in the Over-the-Horizon Radar System Program Office at Hanscom Air Force Base, Massachusetts. From 1980–1984, he worked on low sidelobe phased array and adaptive array research at the Electromagnetic Sciences Division of Rome Air Development Center at Hanscom AFB. From 1987 to present, Dr. Haupt has been teaching at the USAF Academy, Colorado, where he is presently Communications Division Chief and Associate Professor of Electrical Engineering.

Dr. Haupt's research interests include electromagnetic, numerical methods, and chaos theory. Dr. Haupt is a registered Professional Engineer in Colorado, a member of Tau Beta Pi, and a member of URSI Commission B.

Sue Ellen Haupt is a visiting scientist with the National Center for Atmospheric Research (NCAR). Her research interests include numerical methods, fluid mechanics, wave mechanics, and nonlinear dynamics. Dr. Haupt earned her Ph.D. in Atmospheric Science from The University of Michigan. She also holds an M.S. in Mechanical Engineering from Worcester Polytechnic Institute, an M.S. in Engineering Management from Western New England College, and a B.S. in Meteorology from The Pennsylvania State University. Dr. Haupt was a Postdoctoral Fellow at NCAR and a Research Associate with the Program in Applied Mathematics at the University of Colorado. She has been elected to Tau Beta, Pi, Mu Epsilon, and Chi Epsilon Pi.