Multi-Level Adaptive Solutions to Boundary-Value Problems

By Achi Brandt

Abstract. The boundary-value problem is discretized on a set of grids (or finite-element spaces) of widely different mesh sizes. Interaction between these grids (or finite-element spaces) enables us (i) to solve the possibly non-linear system of discrete equations in $O(n)$ operations (40n additions and subtractions for Poisson problems); (ii) to conveniently adapt the discretization (the local mesh size, local order of approximation, etc.) to the evolving solution in an easily optimal way, obtaining "order" approximations and low it, even when singularities are present. General theoretical analysis of the numerical process. Numerical experiments with linear and nonlinear, elliptic and mixed-type (transonic flow) problems confirm the theoretical predictions. Similar techniques for initial-value problems are briefly discussed.

Introduction. In most numerical procedures for solving partial differential equations, the analyst first discretizes the problem, choosing approximating algebraic equations on a finite-dimensional approximations space, and then devises a numerical process to solve the resulting discrete problem. Usually, no real interaction is allowed between discretization and solution processes. This results in a tremendous waste: The discretization process, being unable to predict the proper resolution and the proper order of approximation at each location, produces a mesh which is too fine. The algebraic system thus becomes unnecessarily large in size, while accuracy usually remains rather low, since local smoothness of the solution is not properly exploited.

On the other hand, the solution process often takes advantage of the fact that the algebraic system to be solved is an approximation to continuous equations, and therefore can itself be similarly approximated by other (much simpler) algebraic systems.

The purpose of the work reported here is to study how to intermix discretization and solution processes, thereby making both of them orders-of-magnitude more effective. The method to be proposed is not "saturated," that is, accuracy grows indefinitely as computations proceed. The rate of convergence (overall error as function of computational work $W$) is in principle of "infinite order," e.g., $E \sim \exp(-W)$ for an $n$-dimensional problem which has as solution with scale ratios $>0$; or $E \sim \exp(-W)$, for problems with arbitrary thin layers (see Section 9).

The basic idea of the Multi-Level Adaptive Techniques (MLAT) is to work not

MATHEMATICS OF COMPUTATION, VOLUME 31, NUMBER 138

The article appears on page 333-390.
The presentation in this article is mainly in terms of finite-difference solutions to multi-grid processing and adaptive discretization can be used independently of (2). Adaptive discretization. Mesh-sizes, orders of approximation and other displacements are discussed in Appendix D. Section 10 presents historical nose and acknowledges sections 8.1 and 8.3. Extension to initial-value problems to finite-element discretizations. The latter is briefly discussed in Sections A.5 and with all these pieces naturally patched together by the multi-grid process (Section 7), facilitating, for example, the use of high-order approximations near pieces of boundary, the linear, localized subgrids may be defined in terms of suitable local coordinates. Thus, difference equations should only be constructed on equidistant points. This refinement. In this structure, the difference equations can be defined separately on each of the uniform subgrids, interacting with each other through the multi-grid process. Uniform subgrids, the same sequence used in the multi-grid process, but where the finer levels may be confined to increasingly smaller subdomain to produce the desired local refinement. In this way, the multi-grid process can be viewed as producing an adaptive grid as a sequence of ideas used and relates similar concepts, similar data structures, etc. In particular, an algorithm is suggested for solving discrete equations on the non-uniform grid typically produced by the multi-grid process. The multi-grid process can be viewed as an adaptive grid as a sequence of ideas used and relates similar concepts, similar data structures, etc. In particular, an algorithm is suggested for solving discrete equations on the non-uniform grid typically produced by the multi-grid process. The multi-grid process can be viewed as an adaptive grid as a sequence of ideas used and relates similar concepts, similar data structures, etc. In particular, an algorithm is suggested for solving discrete equations on the non-uniform grid typically produced by the multi-grid process.
Multi-grid philosophy. Suppose we have a set of grids \( G_0, G_1, \ldots, G_n \) approximating the same domain \( U \) with coarser mesh sizes. For simplicity, assume \( G_0 \) is the finest grid and \( G_n \) is the coarsest. The grid size ratio is given by \( h_0 > h_1 > \ldots > h_n \). The main idea is to exploit the fact that the solution on a coarser grid is a good approximation to the solution on the next finer grid. The main steps in the multi-grid algorithm are as follows:

1. **Solve on the finest grid**: Solve the problem on the finest grid \( G_0 \).
2. **Interpolate to a coarser grid**: Interpolate the solution from \( G_0 \) to \( G_1 \).
3. **Solve on the coarser grid**: Solve the problem on the coarser grid \( G_1 \).
4. **Correct on the finest grid**: Use the solution from \( G_1 \) to correct the solution on \( G_0 \).

This process is repeated until the finest grid is reached. The advantage of this approach is that the coarse grids are solved more quickly than the fine grids. The multi-grid algorithm is particularly effective for problems with highly oscillatory solutions.
Achi Brandt
MULTILEVEL ADAPTIVE SOLUTIONS TO BOUNDARY-VALUE PROBLEMS

In the context of solving boundary-value problems, certain residual solutions are not visible on the finest grid and therefore cannot be approximated by coarser grids. Such rapidly-fluctuating residuals are exactly what we get when the approximation itself has been obtained as an interpolation from a coarser-grids solution.

An effective way to damp rapid fluctuations in residuals is by usual relaxation procedures, e.g., the Gauss-Seidel relaxation (see Section 3). At the first few iterations, such procedures usually seem to have fast convergence, with residuals (or corrections) rapidly decreasing from one iteration to the next, but soonafter the convergence rate levels off and becomes very slow. Closer examination (see Section 3 below) shows that the convergence is fast as long as the residuals have strong fluctuations on the scale of the grid. As soon as the residuals are smoothed out, convergence slows down.

This is the point where relaxation sweeps should be discontinued and approximations of the smoothed-out residuals should be employed. The Multi-Grid (MG) methods are systematic methods of mixing relaxation sweeps with approximations of residuals on coarser grids. The residuals on the coarsest grid are coarse enough to make the solution of its algebraic system inexpensive compared with, say, one relaxation sweep over the finest grid.

The following sections further explain these ideas. Section 3.1 explains, through a simple example, what a relaxation sweep is and how it indeed smooths out the residuals very efficiently. The smoothing rates of general difference systems are summarized in Section 3.2. A full Multi-Grid Algorithm, composed of relaxation sweeps over the various grids with suitable interpolations in between, is then presented in Section 4. An important modification for nonlinear problems is described in Section 5 (and used later as the basic algorithm for nonuniform grids and adaptive procedures).

Appendix A supplements these ideas with suitable stopping criteria, details of the interpolation procedures, and special techniques (partial relaxation).

3. Relaxation and its Smoothing Rate.

3.1. An Example. Suppose, for example, we are interested in solving the partial differential equation

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} - \frac{\partial u}{\partial t} = f(x, y) \]

with some suitable boundary conditions. Denoting by \( U^k \) and \( F^k \) approximations of \( U \) and \( f \), respectively, on the grid \( G_k \), the usual second-order discretization of (3.1) is

\[ \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta t} + \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta x} + \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta y} = F^k_{i,j} \]

where \( \Delta x = \Delta y = \frac{1}{2} \Delta t \).

In the context of solving boundary-value problems, certain residual solutions are not visible on the finest grid and therefore cannot be approximated by coarser grids. Such rapidly-fluctuating residuals are exactly what we get when the approximation itself has been obtained as an interpolation from a coarser-grids solution.

An effective way to damp rapid fluctuations in residuals is by usual relaxation procedures, e.g., the Gauss-Seidel relaxation (see Section 3). At the first few iterations, such procedures usually seem to have fast convergence, with residuals (or corrections) rapidly decreasing from one iteration to the next. However, after the convergence rate levels off and becomes very slow. Closer examination (see Section 3 below) shows that the convergence is fast as long as the residuals have strong fluctuations on the scale of the grid. As soon as the residuals are smoothed out, convergence slows down.

This is the point where relaxation sweeps should be discontinued and approximations of the smoothed-out residuals should be employed. The Multi-Grid (MG) methods are systematic methods of mixing relaxation sweeps with approximations of residuals on coarser grids. The residuals on the coarsest grid are coarse enough to make the solution of its algebraic system inexpensive compared with, say, one relaxation sweep over the finest grid.

The following sections further explain these ideas. Section 3.1 explains, through a simple example, what a relaxation sweep is and how it indeed smooths out the residuals very efficiently. The smoothing rates of general difference systems are summarized in Section 3.2. A full Multi-Grid Algorithm, composed of relaxation sweeps over the various grids with suitable interpolations in between, is then presented in Section 4. An important modification for nonlinear problems is described in Section 5 (and used later as the basic algorithm for nonuniform grids and adaptive procedures).

Appendix A supplements these ideas with suitable stopping criteria, details of the interpolation procedures, and special techniques (partial relaxation).

3. Relaxation and its Smoothing Rate.

3.1. An Example. Suppose, for example, we are interested in solving the partial differential equation

\[ \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} + \frac{\partial u}{\partial x} + \frac{\partial u}{\partial y} - \frac{\partial u}{\partial t} = f(x, y) \]

with some suitable boundary conditions. Denoting by \( U^k \) and \( F^k \) approximations of \( U \) and \( f \), respectively, on the grid \( G_k \), the usual second-order discretization of (3.1) is

\[ \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta t} + \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta x} + \frac{u^{k+1}_{i,j} - u^k_{i,j}}{\Delta y} = F^k_{i,j} \]

where \( \Delta x = \Delta y = \frac{1}{2} \Delta t \).
Define \( p(\theta) = \frac{|g(\theta)|^2}{\log(1 + 2|g(\theta)|^2)} \). In the domain of diameter \( 2\alpha \), the lowest Fourier components have a range \( \log(1 + 2|g(\theta)|^2) \). The convergence rate is \( p(\theta) \equiv 1 \), which is the usual ratio (cf. Section 6.2).}

3.2. Geometric Results. The above situation is very general (see [4] and Chapter 3 of [13]). For any uniformly elliptic system of difference equations, it can be shown that if free relaxation sweeps are enough to reduce the high-frequency error components, then a system of equations can be solved, as in the example above, by a simple calculation showing that three relaxation sweeps reduce the high-frequency error components by an order of magnitude.

3.3. Acceleration by Weighting. The smoothing factor \( p(\theta) \) may be improved by various parameters introduced into the scheme. Since \( p(\theta) \) is obtained from \( p(\theta) = \frac{\log(1 + 2|g(\theta)|^2)}{\log(1 + 2|g(\theta)|^2)} \), which is very satisfactory even in the degenerate case \( \delta < 1 \). For linear problems, such parameters can be determined once and for all. In the case of digital computers, the required correction \( u_0 = u_0 - \frac{1}{2} \), and the latter should therefore be preferred, except when parallel implementations are used.

In hyperbolic regions, immediate relaxation schemes are used, since the overall smoothing rate is \( p(\theta) = \frac{\log(1 + 2|g(\theta)|^2)}{\log(1 + 2|g(\theta)|^2)} \), which is very satisfactory, even in the degenerate case. For non-linear problems, it is only in the domain of dependence that the smoothing factor \( p(\theta) \) is obtained. In the case of digital computers, the required correction \( u_0 = u_0 - \frac{1}{2} \), and the latter should therefore be preferred, except when parallel implementations are used.

3.3. Acceleration by Weighting. The smoothing factor \( p(\theta) \) may be improved by various parameters introduced into the scheme. Since \( p(\theta) \) is obtained from \( p(\theta) = \frac{\log(1 + 2|g(\theta)|^2)}{\log(1 + 2|g(\theta)|^2)} \), which is very satisfactory even in the degenerate case \( \delta < 1 \). For linear problems, such parameters can be determined once and for all. In the case of digital computers, the required correction \( u_0 = u_0 - \frac{1}{2} \), and the latter should therefore be preferred, except when parallel implementations are used.

In hyperbolic regions, immediate relaxation schemes are used, since the overall smoothing rate is \( p(\theta) = \frac{\log(1 + 2|g(\theta)|^2)}{\log(1 + 2|g(\theta)|^2)} \), which is very satisfactory, even in the degenerate case. For non-linear problems, it is only in the domain of dependence that the smoothing factor \( p(\theta) \) is obtained. In the case of digital computers, the required correction \( u_0 = u_0 - \frac{1}{2} \), and the latter should therefore be preferred, except when parallel implementations are used.

3.3. Acceleration by Weighting. The smoothing factor \( p(\theta) \) may be improved by various parameters introduced into the scheme. Since \( p(\theta) \) is obtained from \( p(\theta) = \frac{\log(1 + 2|g(\theta)|^2)}{\log(1 + 2|g(\theta)|^2)} \), which is very satisfactory even in the degenerate case \( \delta < 1 \). For linear problems, such parameters can be determined once and for all. In the case of digital computers, the required correction \( u_0 = u_0 - \frac{1}{2} \), and the latter should therefore be preferred, except when parallel implementations are used.
In Section A.7, we show the results of introducing corrections of a different type in the case of differential equations with variable coefficients. For higher accuracy, the procedure is to use a combination of smoothing and grid adaptation. The main advantage of this approach is that it allows for a more efficient solution of the problem, especially for large-scale problems. In the next section, we will explore the use of the FAS (Full Approximation Storage) scheme, which offers an alternative method for solving these types of problems.

Example: In case of incompressible flows, the expression for the pressure is given as follows:

\[ p = \rho \left( \frac{1}{\gamma} \right) \left( \frac{1}{\gamma} \right) \]

The basic rule in Cycle C is that each level is a new level of adaptation on the finest grid. In Cycle C, we introduce a new level of adaptation near the boundaries. In Cycle D, we adapt the levels of adaptation from Cycle C.

The basic idea behind Cycle C is to use a hierarchy of grids, each representing a different level of resolution. By using Cycle C, we can achieve higher accuracy in the solution of the problem while maintaining a reasonable computational cost. The use of Cycle C is particularly useful in problems with large-scale and complex geometries.

In the next section, we will explore the use of the FAS (Full Approximation Storage) scheme, which offers an alternative method for solving these types of problems. The FAS scheme allows for a more efficient solution of the problem, especially for large-scale problems. In the next section, we will explore the use of the FAS (Full Approximation Storage) scheme, which offers an alternative method for solving these types of problems.

Example: In case of incompressible flows, the expression for the pressure is given as follows:

\[ p = \rho \left( \frac{1}{\gamma} \right) \left( \frac{1}{\gamma} \right) \]

The basic rule in Cycle C is that each level is a new level of adaptation on the finest grid. In Cycle C, we introduce a new level of adaptation near the boundaries. In Cycle D, we adapt the levels of adaptation from Cycle C.

The basic idea behind Cycle C is to use a hierarchy of grids, each representing a different level of resolution. By using Cycle C, we can achieve higher accuracy in the solution of the problem while maintaining a reasonable computational cost. The use of Cycle C is particularly useful in problems with large-scale and complex geometries.
Actually requires a coarse-grid correction, i.e., if and when relaxation exhibits a slow rate of convergence. Thus, the equations to be (approximately) satisfied by $u$ are

$$L^*u = f, A^*v = 0, \quad k$$

where $f$ and $v$ are the residuals to the interior equations and the boundary conditions.

We use the notation $I_1$ to represent interpolation from $G_1$ to $G_0$. In case $m < j$, $I_m$ may represent a coarse-grid transfer of values to the coarse grid $G_0$. In the coarse grid $G_0$, $J_m$ may represent a transfer of some modified forms of the fine grid equations and residuals to the coarse grid $G_0$. The stored residuals for the fine grid is only a reduction more than the number.

Then advance $k$ by 1 (to ensure computational at the finest level) and go to step (a).

$$\phi^0 + I_1^0 f + I_1^0 v = f$$

The equations on $G_0$ are thus defined in terms of the approximate solution on $G_1$. The equations on $G_0$ are then derived in terms of the approximate solution on $G_1$. We use the notation $I_1$ to represent interpolation from $G_1$ to $G_0$. In case $m < j$, $I_m$ may represent a coarse-grid transfer of values to the coarse grid $G_0$. In the coarse grid $G_0$, $J_m$ may represent a transfer of some modified forms of the fine grid equations and residuals to the coarse grid $G_0$. The stored residuals for the fine grid is only a reduction more than the number.

The error $r = f - L^*u$ is initialized. $J_0 = 0$ (convergence has been obtained on the finest level), the algorithm is terminated.

We use the notation $I_1$ to represent interpolation from $G_1$ to $G_0$. In case $m < j$, $I_m$ may represent a coarse-grid transfer of values to the coarse grid $G_0$. In the coarse grid $G_0$, $J_m$ may represent a transfer of some modified forms of the fine grid equations and residuals to the coarse grid $G_0$. The stored residuals for the fine grid is only a reduction more than the number.

Then advance $k$ by 1 (to ensure computational at the finest level) and go to step (a).

$$\phi^0 + I_1^0 f + I_1^0 v = f$$

Figure 2. Cycles A and B, Linear Problems

\[ f = I_1^0 r + I_1^0 v + I_1^0 w \]

\[ f = I_1^0 r + I_1^0 v + I_1^0 w \]
346

ACHIBRANDT

4.1. Indefinite Problems and the Size of the Coarsest Grid. If, on any grid $G_k$, the boundary-value problem (4.1) is an indefinite elliptic problem, with eigenvalues (4.7)

and with the corresponding eigenfunctions $V', V, V_1, \ldots, V_n$ then it cannot be solved by straight relaxation. Any relaxation sweep will reduce the error components in the spaces spanned by $V', V_2, \ldots$, but will magnify all error components in the span of $V, V_1, \ldots$. A multi-grid solution, however, is not seriously affected by this magnification, provided the magnified components are suitably reduced by the coarse-grid corrections. This will usually be the case, since these components are basically of low frequency and are well approximated on coarser grids. But care should be taken regarding the coarsest grid:

On the coarsest grid, an indefinite problem should be solved directly (i.e., not by relaxation of any kind). Semi-iterative solutions, like Newton iterations for nonlinear problems, are, of course, permissible. Furthermore, this grid should be fine enough to provide a rough approximation to the $[\nu, (I+\delta)^{-1}]$ eigenfunctions, so that

(4.8) $0 < \mu''(\nu) (1+\delta)[I+\delta]^{-1} < 1$.

where $\mu''(\nu)$ is the symbol of $L''(\nu)$ (see (A.3) in Appendix A) and $b(\nu)$ is its central coefficient.

Usually, $G_0$ can still be coarse enough to have the direct solution of its equations still far less expensive than, say, one relaxation sweep over the finest grid, so that the indefinite problem is solved with the same overall efficiency as definite problems.

5. The FAS (Full Approximation Storage) Algorithm. In the FAS mode of the multi-grid algorithms, instead of storing a correction $\nu(I+\delta)$ (designed to correct the fine-level approximation $u(I+\delta)$), the idea is to store the full current approximation $u(I+\lambda)$, which is the sum of the correction $\nu(I+\delta)$ and its base approximation $u(I+\lambda)$:

(5.1) $u(I+\lambda) = u(I+\lambda) + \nu(I+\delta)$.

In terms of these full-approximation functions, we can write the correction equation

(5.2) $La (\nu(I+\delta)) + \nu(I+\delta) + \nu(I+\delta) = u(I+\lambda)$

where $\nu''(\nu(I+\delta)) = \nu''(\nu) + (1+\delta)[I+\delta]^{-1} \nu''(\nu)$. 

and where $B(I+\lambda)$ is the coarsest grid.

(5.3) $B(I+\lambda) = B(I+\lambda) + \nu(I+\delta)$.

The FAS (Cycle C) algorithm is the same algorithm as in Section 4, with the FAS equations (5.2)—(5.4) replacing (4.1)—(4.3), and with (5.6) replacing (4.5). It is flowchartcd in Figure 3.

The FAS mode has several important advantages: It is suitable for general nonlinear problems, with the same procedures (relaxation and interpolation routines) used at all levels. Thus, for example, only one relaxation routine need be written. Moreover, this mode is suitable for composite grids (nonuniform grids created by increasing finely-grained levels being defined on increasingly smaller subdomains; see Section 7.2), which is the basis for grid adaptation on one hand, and segmental refinement (see Section 7.5) on the other hand.

Generally speaking, the basic feature of the FAS mode is that the function stored on a coarse grid coincides therewith the fine-grids solution; $\nu(I+\delta)$.

This enables us to manipulate accurate solutions on coarser grids, which is a considerable advantage over solutions on coarser grids that are approximate only.
... putting \( P_J - \tilde{P}_J \) = \( P_J \) into the former. (Correction Stages)...

... and slight� the projection mode to the \( G_m \) truncation error.

... for any \( k < M \) can easily be shown (by induction on \( m \), using (5.2) - (5.3)) that...

... which are exactly the \( G_m \) approximations to the \( G_m \) truncation error.

... (5.3) that

... for linear problems on uniform grids, the CS mode is highly preferable.

... for linear problems on uniform grids, the CS mode is highly preferable.

... as an important feature of the multi-grid method is that, although iterative, its total computational work can be predicted in advance by local...

... and zero order Fourier analysis. Such an analysis, which linearizes and freezes the equations...

... which are important in defining natural stopping criteria (see Section 3.8) and grid

... (Section 8.3). Indeed, for any \( k < M \), it can easily be shown...

... and grid adaptation criteria (Section 8.3). Indeed, for any \( k < M \) it can easily be shown...

... for linear problems on uniform grids, the CS mode is highly preferable.

... for linear problems on uniform grids, the CS mode is highly preferable.

... for linear problems on uniform grids, the CS mode is highly preferable.

6.1. Predictability. An important feature of the multi-grid method is that...

6.2. Multigrid Rates of Convergence. To get a convenient measure of convergence per unit work, we define as our work unit (WU) the computational work in one relaxation sweep over the finest grid \( G_N \). Then the number of operations in such a unit is roughly

... and one interpolation. It is difficult to measure the actual rates of relaxation problems in particular applications, and the interpolation work is about the same as the relaxation work.

... the interpolation work is about the same as the relaxation work.

... to ensure that the results of our computational tests are reliable. The purpose of the work on the linear problems in the multi-grid work is to approximate the behavior of the lowest frequencies, which are hard to estimate in time-dependent problems, while precise stability criteria are difficult to apply to high-frequency components, since they have a short coupling range. Thus, these predictions are in general agreement with our computational tests, although each of the tests has only been verified for a small subset of particular problems.

... we have been able to develop new, more accurate, and reliable computational tools based on a better understanding of the behavior of linear systems.

... the main point here, however, is that they are not rigorous estimates, but usually fail to capture the situation in time-dependent problems, where precise stability criteria are difficult to apply to high-frequency components, since they have a short coupling range. Thus, these predictions are in general agreement with our computational tests, although each of the tests has only been verified for a small subset of particular problems.

... we have been able to develop new, more accurate, and reliable computational tools based on a better understanding of the behavior of linear systems.
The smoothing factor near a boundary is lower than usual, which is not the usual case. Smoothing may be accelerated there by partial relaxation sweeps—cf. Section A.9. Thus a multigrid cycle with a relaxation sweep on each level reduces all error components by the factor $\alpha$. The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 = \frac{\rho - 1}{\rho - 1}.$$ 

Hence, the multigrid convergence factor is

$$\text{(6.2)} \quad \rho^p - 1 > \rho^1 - 1 > \ldots > \rho_2 - 1 > \rho_1,$$

where $\rho^1$ is the relaxation sweep $\rho^1 = \alpha$. The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 < \rho_1.$$ 

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1,$$

The amount of work expended in these sweeps is

$$\text{(6.2)} \quad \rho^p - 1 > \rho_1.$$
The second column specifies the relaxation scheme and the extraction parameter. The third column lists the multi-level factors. The fourth column lists the smoothing and multi-grid convergence factors. The table includes the following: ACHIBRANDT and MATLAB data. The document includes equations and text related to multi-level adaptive solution techniques and their application to boundary value problems.
numbers calculated by Allan S. Goodman at IBM Thomas J. Watson Research Center. A more extensive list is in preparation.

Mesh-Size Ratio Optimization. Examining Table I, and many other unlisted examples, it is evident that the mesh-size ratio $\tau = 2$ is closest to optimal, yielding almost minimal $\xi$ and minimal $\eta$. This ratio is more convenient and more economical in the interpolation process (which are ignored in the above calculations) than any other efficient ratio. In practice, therefore, the ratio $\tau = 2$ should be used, giving also a very desirable standardization.

6.3. Overall Multi-Grid Computational Work. Denote by the computational work (in the above Work Units) required to solve the GM problem (2.2), $W^m_{M_1}$ to the level of its truncation errors (Section A.8). If the problem is first solved on GM to the level $r_w = 1$ and if the correct order of interpolation is used to interpolate the solution to GM (so that unnecessary high frequencies are not excited; Section A.2, and in particular (A.7) for $i = 1$), then the residual so obtained is $O(r_0)$. The computational work required to reduce the residual to $O(r_w)$ is

$$W^m = 2 \log_{r_w} \log_{r_0}.$$

Similarly, we can solve the GM-1 problem with work

$$W^{m_{-1}} = \frac{W^m}{r_w} \log_{r_w} \log_{r_0}.$$

Hence, the computational work with respect to the truncation errors is

$$W_{\text{comp}} = W^m + W^{m_{-1}} = \frac{W^m}{r_w} \log_{r_w} \log_{r_0} + W^m.$$

Similarly, we can solve the GM-1 problem with work

$$W^{m_{-1}} = \frac{W^m}{r_w} \log_{r_w} \log_{r_0}.$$

Hence, the computational work with respect to the truncation errors is

$$W_{\text{comp}} = W^m + W^{m_{-1}} = \frac{W^m}{r_w} \log_{r_w} \log_{r_0} + W^m.$$
For solving nonlinear problems, we consider the following procedure:

1. Initialization: Set the initial guess for the solution.
2. Iteration:
   a. Compute the residual: $\|\mathbf{r}\|_2 = \sqrt{\sum_i (r_i)^2}$
   b. Solve the linear system: $\mathbf{A} \mathbf{x} = \mathbf{r}$
   c. Update the solution: $\mathbf{x} = \mathbf{x} + \mathbf{x}$
   d. Check for convergence: $\|\mathbf{r}\|_2 < \epsilon$

where $\mathbf{A}$ is the coefficient matrix, $\mathbf{r}$ is the residual vector, and $\epsilon$ is the tolerance for convergence.

The output shows a multi-grid convergence factor $\lambda = 0.905$, which is close to and slightly faster than the theoretical value $\lambda = 0.595$ shown in Table I.

Many numerical experiments with various elliptic difference equations in different domains were reported in the article. The experiments included the FORTRAN program and the computer output. The output showed a multi-grid convergence factor $\lambda = 0.905$, which is close to and slightly faster than the theoretical value $\lambda = 0.595$ shown in Table I.

The experiments were performed using the multi-grid method, which demonstrated important features of the method: the rate of convergence was essentially insensitive to several factors, including the shape of the domain, the right-hand side $\mathbf{F}$ (which has some influence only at the first couple of cycles, Section A.2) and the finest mesh size $h$ (except for slight variations when $h$ is large). The experiments indicated that the order of the interpolations should be the order of the elliptic equation, as shown in Section A.2 below.

More numerical experiments are now being conducted at the Weizmann Institute of Science in Israel and at IBM Research Center in New York, and will be reported elsewhere in this article.


These experiments were performed in a comparative study of the multi-grid method and the monolithic method. We considered the transonic small disturbance equation in conservation form:

$$\left(\frac{K}{r^2-1}\right)\phi + \frac{\partial}{\partial x}\left(\frac{\rho u}{r^2-1}\right) = 0$$

for the velocity disturbance potential $\phi$ outside an airfoil. Here $K = (r-1)/(r+1)$, $M_*$ is the free-stream Mach number, and $r = (r_1)^2$ is the ratio of specific heats. $\rho_*$ is the airfoil thickness ratio, assumed to be small. Convergence is obtained when the coordinate is stretched. The airfoil, insuitably scaled coordinates, is located at $x = 0$, and we consider non-homogeneous boundary conditions:

$$\phi = 0 \quad \text{for} \quad x < 0,$$

$$\phi = F'(x) \quad \text{for} \quad x > 0,$$

where $F'(x)$ is the airfoil thickness function which we took to be parabolic. Equation (6.7) is of hyperbolic type depending on whether $\frac{K}{r^2-1} < \epsilon$ or $\frac{K}{r^2-1} > \epsilon$, where $\epsilon$ is the artificial viscosity function which we take to be parabolic. The artificial viscosity function is given by:

$$\phi = \begin{cases} 0 & \text{if} \quad \frac{K}{r^2-1} < \epsilon \\ \phi & \text{if} \quad \frac{K}{r^2-1} > \epsilon \end{cases}$$

The value of the artificial viscosity function is then $\phi = \phi + 0 + (\phi + 0 + (\cdots))^\infty$, where $\phi$ is the number of points in $\phi$. The total number of operations is then $\phi + 0 + (\phi + 0 + (\cdots))^\infty$. For solving the problem, we use the following procedure:

1. Initialization: Set the initial guess for the solution.
2. Iteration:
   a. Compute the residual: $\|\mathbf{r}\|_2 = \sqrt{\sum_i (r_i)^2}$
   b. Solve the linear system: $\mathbf{A} \mathbf{x} = \mathbf{r}$
   c. Update the solution: $\mathbf{x} = \mathbf{x} + \mathbf{x}$
   d. Check for convergence: $\|\mathbf{r}\|_2 < \epsilon$

where $\mathbf{A}$ is the coefficient matrix, $\mathbf{r}$ is the residual vector, and $\epsilon$ is the tolerance for convergence.
Thus, be it in the original or in some transformed domain, one would like to have a convenient system for local refinements, with minimal bookkeeping and efficient storage for keeping coefficients, or additional work in re-computing them every sweep.

Another approach for organizing a nonuniform grid is by coordinate transformation. The transformed difference equations (stretching only the x-coordinate) and indeed faster the convergence of the solution process remains practically as before (Section 5), with similar efficiency. Mod discussed in Section 5 was programmed both in the CS (Section 4) and the FAS (Section 5) modes, with vertical (y) linear relaxation, marching in the stream direction. The multi-grid solution on a completely general grid (see, however, Sections 7.3 and A.5), and complete generality is not necessary for obtaining any desired refinement pattern.

Calculation of the finer levels is defined in terms of suitable local coordinates. The multi-grid FAS solution on a fully general grid processes in different domains. Special refinement of the grid is realized near singular points, etc. A general method for locally choosing mesh-sizes and approximation orders is described in Section 8. An important feature of the method is adaptivity: the grid may change during the solution process. The difference equations we used were essentially the Murman's conservative scheme (191; for a recent account of solution methods, see 181), where the main idea was to adaptively use upwind differencing in the hyperbolic region and central differencing in the elliptic region, keeping the system conservative. For relaxation we used the usual line relaxation for actual computations. There are two popular approaches: one, usually used with the coarse grid, is a method which employs the grid organization for "segmental refinement", a multi-grid process. The other approach is to let the sequence of uniform grids $G_0, G_1, \ldots, G_M$ (cf. Section 2) be open-ended and noncoextensive (i.e., finer levels may be introduced on increasingly wider domains to cover unbounded domains), and further let each grid be divided into a number of subdomains, and let these subdomains be further divided into smaller subdomains, etc. The actual convergence factors, observed in our experiments with moderately supercritical flows ($M = 0.7$ and $M = 0.85$, $\alpha = 0.1$) on a $64 \times 32$ grid, were 0.52 to 0.53, just slightly faster than the theoretical value. (See detailed output in 1121.)

The work shown in [12] is slightly different, containing the work in the present paper (19). It is past History whose re-statement was delayed until 19. [5] = 0.45 and $\alpha = 0.45$ and $\alpha = 0.55$. The elliptic equations were solved in each domain with modest iteration (stretching only the z-coordinate), and the results restored the convergence rate of the full half-plane, gaving difference equations that again exhibited slow multi-grid convergence. This, too, is explainable by the mode analysis. For example, in the transition region between the elliptic (subsonic) points, and $\alpha = 0$ at supersonic points.

For highly supercritical flows ($M = 0.95$ and $\alpha = 0.1$) the MG convergence rate deteriorated, although it was still three times faster than solution by line relaxation. For switching to a coarse grid in a transonic region, $\alpha = 0$ at subsonic points, we were interested in cases where $K < I$ and $\alpha = 0$, and hence, in smooth elliptic regions ($\alpha > 0$) with $\alpha = 0$ and $\alpha = 0$ in smooth elliptic regions far from shock waves. The work presented here was carried out by one of the authors. We expect the slow convergence rates to be improved with future work in this direction. The work here in [12] is slightly different, containing the work in the present paper (19). It is past History whose re-statement was delayed until 19. [5] = 0.45 and $\alpha = 0.45$ and $\alpha = 0.55$. The elliptic equations were solved in each domain with modest iteration (stretching only the z-coordinate), and the results restored the convergence rate of the full half-plane, gaving difference equations that again exhibited slow multi-grid convergence. This, too, is explainable by the mode analysis. For example, in the transition region between the elliptic (subsonic) points, and $\alpha = 0$ at supersonic points.

For highly supercritical flows ($M = 0.95$ and $\alpha = 0.1$) the MG convergence rate deteriorated, although it was still three times faster than solution by line relaxation. For switching to a coarse grid in a transonic region, $\alpha = 0$ at subsonic points, we were interested in cases where $K < I$ and $\alpha = 0$, and hence, in smooth elliptic regions ($\alpha > 0$) with $\alpha = 0$ and $\alpha = 0$ in smooth elliptic regions far from shock waves. The work presented here was carried out by one of the authors. We expect the slow convergence rates to be improved with future work in this direction. The work here in [12] is slightly different, containing the work in the present paper (19). It is past History whose re-statement was delayed until 19. [5] = 0.45 and $\alpha = 0.45$ and $\alpha = 0.55$. The elliptic equations were solved in each domain with modest iteration (stretching only the z-coordinate), and the results restored the convergence rate of the full half-plane, gaving difference equations that again exhibited slow multi-grid convergence. This, too, is explainable by the mode analysis. For example, in the transition region between the elliptic (subsonic) points, and $\alpha = 0$ at supersonic points.
problem on a bounded domain or global domain can be extended (or contracted) using special interpolation (or extrapolation) routines tailored for easy changes.

For example, each grid point is used (or counted as a point of a function value) only once by some special loop over grid points. Thus, the system of difference equations can be employed on a global domain or global domain can be modified (or contracted) using special interpolation (or extrapolation) routines tailored for easy changes.

For example, each grid point is used (or counted as a point of a function value) only once by some special loop over grid points. Thus, the system of difference equations can be employed on a global domain or global domain can be modified (or contracted) using special interpolation (or extrapolation) routines tailored for easy changes.

For example, each grid point is used (or counted as a point of a function value) only once by some special loop over grid points. Thus, the system of difference equations can be employed on a global domain or global domain can be modified (or contracted) using special interpolation (or extrapolation) routines tailored for easy changes.
A multi-grid method for patching together a collection of local grids $G_1, G_2, \ldots$

### 7.4 Local Transformations

Another important generalization of the above can be given to a collection of local grids $G_1, G_2, \ldots$. Here, in principle, the rest may in principle be as be.

The multi-grid algorithm is the same as in Section 7.2. The discrete equations of the form $a_k = A_k u_k$ can be solved on each grid level, using an iterative method. In addition, the solution on each grid level is obtained through its aggregation into a single solution on the next coarse grid.

### 7.3 Finite-Element Generalization

The structure and solution process outlined above for the potential of the problem.

The multi-grid method is a very powerful tool for solving the system of linear equations obtained from discretization of the problem. It is particularly effective for problems with large solution spaces and can significantly reduce the computational effort required.

### 7.2 Theoretical Foundation

The theoretical foundation of the multi-grid method is based on the use of fine-grained approximations to coarse-grained problems. This approach allows for the efficient solution of large-scale problems through the use of iterative methods.

### 7.1 Methodology

The methodology for implementing the multi-grid method involves the following steps:

1. **Pre-solve:** Solve the system on the finest grid.
2. **Coarse-graining:** Coarsen the grid by combining adjacent grids.
3. **Correct:** Solve the system on the coarser grid and then refine the solution back to the finer grid.
4. **Iterate:** Repeat the process until convergence is achieved.

### 7.4 Local Transformations

Another important generalization of the above can be given to a collection of local grids $G_1, G_2, \ldots$. Here, in principle, the rest may in principle be as be.

The multi-grid algorithm is the same as in Section 7.2. The discrete equations of the form $a_k = A_k u_k$ can be solved on each grid level, using an iterative method. In addition, the solution on each grid level is obtained through its aggregation into a single solution on the next coarse grid.

### 7.3 Finite-Element Generalization

The structure and solution process outlined above for the potential of the problem.

The multi-grid method is a very powerful tool for solving the system of linear equations obtained from discretization of the problem. It is particularly effective for problems with large solution spaces and can significantly reduce the computational effort required.

### 7.2 Theoretical Foundation

The theoretical foundation of the multi-grid method is based on the use of fine-grained approximations to coarse-grained problems. This approach allows for the efficient solution of large-scale problems through the use of iterative methods.

### 7.1 Methodology

The methodology for implementing the multi-grid method involves the following steps:

1. **Pre-solve:** Solve the system on the finest grid.
2. **Coarse-graining:** Coarsen the grid by combining adjacent grids.
3. **Correct:** Solve the system on the coarser grid and then refine the solution back to the finer grid.
4. **Iterate:** Repeat the process until convergence is achieved.
The error estimator $E$ is a functional that estimates, for any given numerical approximation $L'$, its distance to the true solution $L$. This is necessary and justifiable assumption; see Section 8.2, so that $E$ can be written as a linear functional.

In principle, such a functional should be furnished whenever a problem is submitted for numerical solution; in practice, it is seldom provided. To have such an estimator, i.e., an idea about what error norm we intend to minimize, given the goal, even roughly, we can usually formulate quite easily. We assume that the numerical approximation $L'$ is in some suitable neighborhood of the true solution $L$, i.e., the numerical error $|L - L'|$ is small, say, less than $h^p$ for some $p > 0$. This ensures that the estimator $E(L')$ is small, say, less than $h^{p-1}$ for the same $p$. We denote this error by $e(T, U)$, where $T$ is the computational domain and $U$ is the solution of the differential boundary-value problem.

Having the control work much smaller than the actual numerical work, using the optimization problem only as a loose directive for sensible discretization, is not our purpose, is enormously harder and, in fact, is self-defeating, since it requires too much computational work to be invested in controlling hand-picking. We will aim at obtaining $E$ which is one or two orders of magnitude smaller than the minimum needed. Note below that $\log(1/E)$ is usually proportional to $h^p$. Full optimization (or, equivalently, to invest work $W$ which is by some fraction more than theoretically necessary) means that we want to minimize both $E$ and $W$, but their rate of exchange. It is important, however, to promptly realize that $W$ is not our concern; it is enough, for instance, to obtain $E$ which is one or two orders of magnitude larger than the minimum necessary, to emphasize that we should not take this optimization to pedantically: it is enough, for instance, to take the $W$ which is by some fraction more than theoretically necessary. For optimization, we should not focus on the error $E$ itself, but on the cost $W$.

The correct forcing function can be computed by segmental refinement.

"Segmental refinement" is the refinement of one subdomain at a time. To see how this can be done, consider the local difference equation on a coarser grid $G_k$ of a domain $Q$ whose solution $u_k$ is known. The forcing functions $F_k$ on $Q_k$ differ from the forcing functions $F_k$ on $Q$. The correct forcing function $F_k$ can be obtained by refining only a few neighboring grid points, i.e., those grid points where $h$ is the finest mesh-size and $R$ is the diameter of the domain. The local difference equation on $Q_k$ is then obtained on all grids.

The correct forcing function $F_k$ can be obtained by refining only a few neighboring grid points, i.e., those grid points where $h$ is the finest mesh-size and $R$ is the diameter of the domain. The local difference equation on $Q_k$ is then obtained on all grids.

The correct forcing function $F_k$ can be obtained by refining only a few neighboring grid points, i.e., those grid points where $h$ is the finest mesh-size and $R$ is the diameter of the domain. The local difference equation on $Q_k$ is then obtained on all grids.

"Segmental refinement" is the refinement of one subdomain at a time. To see how this can be done, consider the local difference equation on a coarser grid $G_k$ of a domain $Q$ whose solution $u_k$ is known. The forcing functions $F_k$ on $Q_k$ differ from the forcing functions $F_k$ on $Q$. The correct forcing function $F_k$ can be obtained by refining only a few neighboring grid points, i.e., those grid points where $h$ is the finest mesh-size and $R$ is the diameter of the domain. The local difference equation on $Q_k$ is then obtained on all grids.

The correct forcing function $F_k$ can be obtained by refining only a few neighboring grid points, i.e., those grid points where $h$ is the finest mesh-size and $R$ is the diameter of the domain. The local difference equation on $Q_k$ is then obtained on all grids.
Emphasizing the meaning of \( (8.5) \) is that we cannot lower \( E \) by trading work (e.g., by taking smaller \( i \) at one point and larger at another, keeping \( p \) constant, or trading a change in \( i \) with a change in \( p \)).

Equations \( (8.5) \) makes some essential simplifications in the optimization problem:
- They regard \( i \) and \( p \) as defined at all points \( x \in \Omega \).
- \( h \) and \( p \) are assumed to be continuous variables, whereas in practice they are discrete (\( p \) should be a positive integer).
- \( i \) and \( p \) are restricted by some numerical considerations. (\( i \) and \( p \) are restricted to some numerical considerations.)

These simplifications are crucial for our approach.

We get the following equations as points of the optimization problem:

\[
\begin{align*}
(x) \frac{d}{dx} \left( \frac{x}{2} \right) + \lambda(x) &= 0, \\
(x) \frac{d}{dx} \left( \frac{x}{2} \right) &= 0.
\end{align*}
\]

Equations \( (8.5) \) are continuous equations. Substituting \( (8.1) \) and \( (8.4) \) into \( (8.5a) \) and \( (8.5b) \), we get the following equations at each point of the optimization problem:

\[
\begin{align*}
\frac{M_i}{\lambda} - p &= 0, \\
\frac{M_i}{\lambda} + \lambda &= 0.
\end{align*}
\]

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The work functional \( W \) is the functional of minimizing \( E \) for fixed \( i \) and \( p \).

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.

The choice of \( \lambda \) can be made in order to minimize \( \lambda \) by multiplication of the equation.
The practice of discretization control

The main practical restrictions imposed in changing \( y \) make it advisable to keep the residuals as smooth as possible, since a continuation process is necessary, in principle, not only for nonlinear problems, but also for linear problems.

In our system, a continuation process with crude accuracy and little work is usually obtained by selecting a large value for the control parameter \( A \) (see Section 7.2). Then, in the final step \( (y = 8) \), a step-size \( 6^7 \) is sharply decreased as \( y \) is increased, and \( 6^7 \) is kept constant for the remainder of the solution.

The multi-grid discretization method outlined in Section 7.2 is particularly useful in this respect, since the multi-grid process may be viewed as a multi-grid process of solution control by the two unknowns, where the control parameter \( A \) is varied.

In the theoretical discretization equations (8.9) the unknown, the discretized problem is nonlinear, even if the differential problem is linear. Thus, although sometimes we may get away without a continuation process, in principle a continuation process must be employed (either because as starting solution is "close enough" so that the solution can be obtained easily). The continuation process is not a waste, for several reasons. In many cases, the computational work for additional accuracy (see (8.6)) is normally done in the error process to ensure that the approximation is within the desired accuracy. Thus, the multi-grid process (see (5.7), for example) gives us local estimates of what is the true solution. In the presence of coupling, the problem is to be solved, \( P(y_0) \) is easily separable, \( (6.9) \) and \( (5.7) \), and \( (6.10) \) are the target \( (6.2) \). Where \( P(\ell) \) and \( \ell \) are the solutions to \( P(\ell) \), \( \ell \) is the target solution, and \( P(\ell) \) is the target problem. Usually, \( y \) is some natural physical parameter (the Reynolds number, the Mach number, etc.) expressed in terms of which the solution is obtained. Thus, although sometimes we may get away without a continuation process, in principle a continuation process must be employed (either because as starting solution is "close enough" so that the solution can be obtained easily).

The multi-grid discretization method outlined in Section 7.2 is particularly useful in this respect, since the multi-grid process may be viewed as a multi-grid process of solution control by the two unknowns, where the control parameter \( A \) is varied. In the theoretical discretization equations (8.9) the unknown, the discretized problem is nonlinear, even if the differential problem is linear. Thus, although sometimes we may get away without a continuation process, in principle a continuation process must be employed (either because as starting solution is "close enough" so that the solution can be obtained easily). The continuation process is not a waste, for several reasons. In many cases, the computational work for additional accuracy (see (8.6)) is normally done in the error process to ensure that the approximation is within the desired accuracy. Thus, the multi-grid process (see (5.7), for example) gives us local estimates of what is the true solution. In the presence of coupling, the problem is to be solved, \( P(y_0) \) is easily separable, \( (6.9) \) and \( (5.7) \), and \( (6.10) \) are the target \( (6.2) \). Where \( P(\ell) \) and \( \ell \) are the solutions to \( P(\ell) \), \( \ell \) is the target solution, and \( P(\ell) \) is the target problem. Usually, \( y \) is some natural physical parameter (the Reynolds number, the Mach number, etc.) expressed in terms of which the solution is obtained. Thus, although sometimes we may get away without a continuation process, in principle a continuation process must be employed (either because as starting solution is "close enough" so that the solution can be obtained easily).
The practical adaptive procedure is proposed to be generally along the following change (e.g., refinement) is introduced only if there is a point where the change is 'overdue' (e.g., where $\tau > 0$). Together with such a point the change is introduced in all neighboring points where the error is 'due' (e.g., where $\tau > 3$). The changed subgrid (denoted $G_{k+1}$ in the above example) becomes an inner point of Section 7.2) in the new subgrid $G_{k+1+1}$. (ii) Holes are filled: that is, if, on any grid line, a couple of points are missing in between grid points, then the missing points are added to the grid. At each point may be used as a local estimate for the decrease in $E$ per unit volume (cf. the control rate of exchange). Hence we compute $\tau$ and $\alpha$, which are the ratio of exchanging accuracy to the main work being in interpolating the approximate solution to the new piece of grid work (cf. Section 7.1). More sophisticated tests may be based on estimating the coarseness of adaptation (e.g., $\tau$ and $\alpha$) for comparison with a known work number (e.g., the work of a subgrid). (iii) Changing the discretization is itself expensive since it is done by extending or contracting uniform grids (cf. Section 7.1). Hence we can only test for changes when the work of changing the discretization is larger than the work of the present change. This is in the context of the following tests: (i) If the work of changing the discretization is larger than the work of the present change, then we change the discretization step $(\tau > 0)$ and return to the next step $(\alpha > 0)$. (ii) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (iii) If the work of changing the discretization is smaller than the work of the present change, then we change the discretization step $(\tau > 0)$ and return to the next step $(\alpha > 0)$. (i) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (ii) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (iii) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (iv) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (v) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (vi) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (vii) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (viii) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (ix) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (x) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (xi) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1). (xii) More sophisticated tests may be based on assuming that the change is produced by some finite work per unit volume (cf. Section 7.1).
Adaptive Discretization: Case Studies. To get a transparent view of the discretization patterns and the accuracy-work relationship typical to the adaptive procedures proposed above, we consider now several test cases which are simple enough to be analyzed in closed forms. That is, we consider problems with known solutions and simple behavior of the local truncation errors, and we calculate the discretization functions $h(x)$ and $p(x)$ that would be selected by the local optimization equations (8.9), and the resulting relation between the error estimator $E$ and the computational work $W$.

9.1. Uniform-Scale Problems. A problem is said to have the uniform scale if $\nu(x) = 1$ for all $x$. We take $\nu = 1$ in some practical models. We begin on no exact information in that case and assume that the second-order term is dominant (corresponding to $\nu = 0$ in Section 9.2). We will also assume (see (8.4))

$$w(p) = w_0 p.$$  

Usually $p = 1$, since the number of terms in the difference equations, and hence also the amount of computer operations at each grid point, are proportional to $p$. A value of $p = 2$ is appropriate if we assume that we have to increase the precision of our arithmetic when we increase $p$. Rescaling $W$, we can assume that $w_0 = 1$.

Using (9.1) in Eqs. (8.9), we get

$$G(x) = \frac{1}{1 - \nu} \frac{d}{dx} \log |\nu|,$$

where

$$\nu(x) = \begin{cases} \frac{d}{dx} \log |\nu| & \text{if } 0 < x < x_0, \\ \frac{d}{dx} \log |\nu| & \text{if } x_0 < x < l. \end{cases}$$

Hence, denoting by $\nu_0$ the value of $\nu$ at that point, we have

$$\nu(x) = \begin{cases} \frac{d}{dx} \log |\nu| & \text{if } 0 < x < x_0, \\ \frac{d}{dx} \log |\nu| & \text{if } x_0 < x < l. \end{cases}$$

If $x_0 = 0$, then (9.1) applies throughout, and hence the condition $x_0 \neq 0$ itself becomes, by (9.1c), (9.12),

$$\frac{d}{dx} \log \nu = \frac{d}{dx} \log \nu_0,$$

which is some sort of uniformity in the difference equations, and hence also in the solution.

$$G(x) = \frac{1}{1 - \nu} \frac{d}{dx} \log |\nu|,$$

where

$$\nu(x) = \begin{cases} \frac{d}{dx} \log |\nu| & \text{if } 0 < x < x_0, \\ \frac{d}{dx} \log |\nu| & \text{if } x_0 < x < l. \end{cases}$$

If the local truncation error is desired, then $\nu$ is the correct choice to use.

9.2. One-Dimensional Case. Consider a problem of the form

$$\frac{d}{dx} \left[ \frac{d}{dx} u(x) \right] = f(x),$$

and the condition $x > 0$ is replaced by $x > x_0$. The solution of this problem is

$$u(x) = \int_{x_0}^x \frac{d}{dx} \left[ \frac{d}{dx} u(y) \right] dy + C,$$

where $C$ is some sort of uniformity in the difference equations, and hence also in the solution.

$$G(x) = \frac{1}{1 - \nu} \frac{d}{dx} \log |\nu|,$$

where

$$\nu(x) = \begin{cases} \frac{d}{dx} \log |\nu| & \text{if } 0 < x < x_0, \\ \frac{d}{dx} \log |\nu| & \text{if } x_0 < x < l. \end{cases}$$

If the local truncation error is desired, then $\nu$ is the correct choice to use.
When \( \epsilon \) is very small, problem (9.8) is singularly perturbed, and its solution has a boundary layer near \( x = 0 \). The above mesh size \( h = \epsilon \) is too small to be practical. Indeed, in the optimal discretization (9.1), for small \( \epsilon \) we get small \( x_0 \), and an "external region" \( x_0 < x < 1 \) is formed where the mesh size grows exponentially from \( r \). The small mesh size is used only to resolve the boundary layer. In this simplified problem, the solution away from the boundary layer (i.e., for \( x > x_0 \)) is practically constant, so that indefinitely large \( h \) is suitable. Usually \( h \) will grow exponentially, as in (9.11b), from \( h = T \delta / C \) to some definite value suitable for the external region. In the transition region we have \( p = 2 \), i.e., the minimal order of differencing \( \delta \) is used in the region where \( h \) changes.

From (9.11) and (9.9) we get for small \( \epsilon \)

\[
\begin{align*}
E & = \int_0^{x_0} \frac{1}{h} \left( \log x \right)^2 dx \\
& = \int_{0}^{x_0} \frac{1}{h} \left( \log x \right)^2 dx
\end{align*}
\]

where the integrals are separately calculated in \( (0, x_0) \) and \( (x_0, 1) \). Thus, \( E \) converges exponentially as a function of \( \Delta t^2 \). Instead of \( \Delta t \), but this rate is independent of \( \tau \) and does not deteriorate as \( \epsilon \to 0 \).

9.4. Singular Perturbation Without Boundary-Layer Resolution
To see the effect of choosing different error weighting functions, consider again the above problem (Sections 9.2, 9.3), but with the choice \( G(x) = x \). This choice is typical to cases where one is not interested in calculating boundary derivatives of the solution (see (8.3)). We then get

\[
\epsilon = \frac{1}{x} \left( \log x \right)^2
\]

Therefore, for small \( \epsilon \) and reasonable \( \delta > 0 \) and \( p = 2 \) for all \( x \). Hence, no resolution of the boundary layer is formed. Indeed, by (9.5b), for very small \( \epsilon \) (singular perturbation case)

\[
\epsilon = \frac{1}{x} \left( \log x \right)^2
\]

so that \( h_0 > 0 \). In the practical situation where the solution in the external region is not constant, the actual mesh size will be determined by the external region.

9.5. Boundary Layers
Consider the two-dimensional Poisson equation \( \nabla^2 U = f \) with smooth \( f \) and homogeneous boundary conditions near a boundary corner with angle \( \theta \), \( 0 < \theta < \pi \). Denoting by \( r \) the distance from the corner, at small \( r \) the solution \( U \) is \( O(r^n) \), and so is also the error weighting function \( G \) if accuracy is sought in the solution but not in its derivatives near the boundary. Hence, \( f = \delta \frac{\partial U}{\partial n} \) is a boundary condition.

In the practical grid organization (Section 8.3) finer levels \( G_k \) with increasingly smaller mesh sizes \( h = 2^{-k} h_0 \) will be introduced near the corner. By (9.19), the level \( G_k \) will extend from the corner to a distance \( r = 2^{-k} r_0 \). Since \( r < 2^{-k} r_0 \), the solution will not be influenced near the corner. By (9.20), if \( \theta = \pi \), the finest mesh size is \( h_k = 2^{-k} h_0 \) for all \( x \). Hence, no resolution of the boundary layer is formed. Indeed, by (9.5b), for very small \( \epsilon \) (singular perturbation case)

\[
\epsilon = \frac{1}{x} \left( \log x \right)^2
\]

so that \( h_0 > 0 \). In the practical situation where the solution in the external region is not constant, the actual mesh size will be determined by the external region.

9.6. Singularities
Like boundary corners, all kinds of other problem singularities, when treated adaptively, cause no degradation of the convergence rate of \( E \). In this section, the problem is only piecewise smooth, or has algebraic singularities.

Consider for example the differential equation \( LU = F \) where \( F \) is smooth except for a jump discontinuity at \( x = 0 \). Whatever the approximation order \( p \), the system will find \( E(\tau) \) to be \( O(1) \) at all points whose difference equation includes values on both sides of the discontinuity. At such points, the approximation order \( p \) will prevail. The discontinuity at \( x = 0 \) where the approximation order \( p \) is smooth but \( f \) is not smooth is called a "weak singularity." In the practical grid organization (Section 8.3) finer levels \( G_k \) with increasingly smaller mesh sizes \( h = 2^{-k} h_0 \) will be introduced near the corner. By (9.19), the level \( G_k \) will extend from the corner to a distance \( r = 2^{-k} r_0 \). Since \( r < 2^{-k} r_0 \), the solution will not be influenced near the corner. By (9.20), if \( \theta = \pi \), the finest mesh size is \( h_k = 2^{-k} h_0 \) for all \( x \). Hence, no resolution of the boundary layer is formed. Indeed, by (9.5b), for very small \( \epsilon \) (singular perturbation case)

\[
\epsilon = \frac{1}{x} \left( \log x \right)^2
\]

so that \( h_0 > 0 \). In the practical situation where the solution in the external region is not constant, the actual mesh size will be determined by the external region.

10. Historical Notes and Acknowledgements
Coarse-grid acceleration techniques were recommended and used by several authors, including Southwell (13,14), Stiefel (15), Fedorenko (151), Ahmed (19), Wachspress (171, Chapter 9), de Is Vallée Poussin (1161), and Settari and Aziz (241). Southwell called his technique "block" and more generally "group relaxation," described it as "almost essential to practical success," and gave heuristic explanations as well as practical implementation methods based on variational considerations ("the aim being to reduce the total energy by as great an amount as possible"). He also depicted procedures of "advance to a finer net" (14). Techniques of multiplicative coarse-grid corrections (special cases of which are called them "perturbation correction") were developed by Wachspress (17, §11) in coarse-grid acceleration techniques or "perturbation correction." This is the usual order of difference equations in the region where they change. This is also the minimum order of difference equations for the extremal region. In the transition region, we have some \( \approx (1) \) to (1) (9.1) (9.2) (9.3) (9.4). In the multiplicity of the coarse-grid acceleration techniques, the extremal coarse equation is used in the region where the finest coarse equation is used. The smallest mesh size in the region where the finest coarse equation is used is \( (1) \) (9.1) (9.2) (9.3) (9.4). However, the coarse grid is then coarse and not so useful. The error contribution from a region of...
Our original approach was to regard the finer levels as "correcting" the coarser level (cf. Sections 3, 7.2, and 7.5 above). For uniform nonadaptive grid this approach turns out to be equivalent to the one implied by 161, but fundamentally it is different and more efficient. Fedorenko [61] mainly for theoretical purposes. Namely, he rigorously proved that for a rectangular grid with $n$ points, by a factor, is $O(n \log n)$. Bakhvalov [1] generalized this result to any second-order elliptic operator with continuous coefficients.

For large $n$, this is the best possible result—except for the actual value of the coefficient. The Fedorenko estimate can be written as

$$W(n, e) \leq W(n, \epsilon_1) \leq \frac{C n}{\log n}.$$
MULTILEVEL ADAPTIVE SOLUTIONS TO BOUNDARY-VALUE PROBLEMS

A. The Coarse-to-Fine Interpolation

On the other hand, it follows from the previous section that the effective smoothing factor per iteration sweep is obtained by solving the coarse-grid correction on the coarsest level of the problem, and then applying the smoothing operator per sweep.

In order to ensure that the high-frequency modes are retained, a lower-order effective smoothing factor is used in the initial stages of solving a problem. The effective smoothing factor is defined as the ratio of the convergence factor for the fine-grid correction to the convergence factor for the coarse-grid correction.

The Basic Rule: The order of interpolation should not exceed the order of the differential equation. In particular, polynomial interpolations should be made with polynomials of degree $m - 1$. Higher order interpolations ($m > 1$) are desirable in the initial stages of solving a problem, where the high-frequency modes are dominant. However, in the later stages of solving a problem, the low-frequency modes are dominant. Therefore, the effective smoothing factor is defined as the ratio of the convergence factor for the fine-grid correction to the convergence factor for the coarse-grid correction.

The Effective Smoothing Rate. The smoothing factor was defined in (3.8) as the largest convergence factor for all components not represented at the coarsest level. More relevant, however, is the largest factor among all components for which the coarse-grid correction is ineffective, namely,

$$\max(p(0) : \omega/2 = \omega_{01} \text{ for all } 01 \text{ where } \omega_{01} = o(0) \text{ for all } 01 \text{ where } \omega_{01} = o(0))$$

where $p$ is the order of the differential equation. The coarse-grid correction must be effective in the later stages of solving a problem. The effective smoothing factor is defined as the ratio of the convergence factor for the fine-grid correction to the convergence factor for the coarse-grid correction. The effective smoothing factor is defined as the ratio of the convergence factor for the fine-grid correction to the convergence factor for the coarse-grid correction. The effective smoothing factor is defined as the ratio of the convergence factor for the fine-grid correction to the convergence factor for the coarse-grid correction.
A.4. The Fine-to-Coarse Weighting of Residuals (1')

and the Coarse-Grid Operator $L'$

The transfer of the $G^4$ errors' first-order corrections $\eta'$ to the coarser grid $G^3$, to serve as the right-hand side $f''$ (see Section 4, Step (c)) can be made in many ways. Generally, $f''_i$ is defined as some weighted average of the residuals $\eta'$ in neighboring $G^3$ points:

\[(A.9) f'(x) = \sum \eta'(x) = \sum \eta'(x + v_i), \quad v_1, v_2, \ldots, v_n \text{ integers, and the summation is over a small set.} \]

In terms of these weights, $p(O)$ in (A.1) is given by

\[(A.10) p(O) = \frac{1}{J} \sum \eta'(x + v_i), \quad v_i \text{ integers.} \]

The coarse-grid operator $L'$ can also be chosen in many ways, e.g., as some weighted average of the operator $L$ in neighboring points.

How are these choices to be made? The main purpose should be to minimize, but without investing too much computational work in the weighting. Usually, it is ...

preferably derived from $L$, its coefficients will automatically be the suitable weighted averages of the coefficient of $L$. (In case $L$ is given and $L'$ is not, and if the coefficients vary rapidly, then the weighting (A.12) below should be used for defining the $L'$ coefficients.)

It is clear from (A.4) that we should take $p(O) = 1$. There is no apriori restriction, however, on the signs of the weights $p$. The trivial weighting (A.11) $p(O) = 1$, $q(O) = 1$, where $q(O)$ is the coefficient of the next point, has an important advantage in saving computations, not only because the weighting itself is saved, but mainly because it requires the computation of $\eta'$ only at the points, while other weightings compute at all $G$ points, an additional work comparable to one $G$ relaxation sweep.

Nontrivial weighting of residuals is especially important for difference equations with rapidly varying coefficients (large variations per mesh-width), and, in particular, for nonlinear equations, especially on coarse grids. In the $G^3$ relaxation sweeps for such equations, the slowly converging low-frequency modes are coupled with high-frequency modes. In the correction function (cf. Sections 2 or 4) these accompanying high-frequency modes are small (i.e., their amplitudes are small relative to the low-frequency amplitudes), and hence $v''_i$ can still be well approximated by a function $v''_i$ on the coarser grid $G^3$. In the residual however, these accompanying high-frequency modes may be large (comparable in amplitude to the low-frequency modes) and their distortion of $\eta''$ will be small only if their contribution to $\eta''_1 f''_i$ is also of high frequency (in $G^3$). This is indeed the case, except for residual components $\eta''_i (0, \ldots, 0)$ where $1_i \neq 0$ and each $1_i$ is close either to 0 or to 1. The weighting of these terms should be

\[(A.12) p(O) = \begin{cases} 1, & \text{if } 0 \leq |i| < 3, \\ 0, & \text{if } |i| \geq 3. \end{cases} \]

Examples. For symmetric second-order equations, injection should usually be used. For the 5-point Laplace operator, for example, if we take to be injection, linear interpolation, and $L''_i$ also a 5-point Laplace operator, we get $c_i = 1$, the minimal possible value. Any weighting is a pure waste, including the "optimal" weighting

\[(A.12a) p(O) = \frac{1}{3}, \quad p(O) = p(O) = 1, \quad p(O) = 0, \quad \text{for } |i| \geq 1. \]

Numerical experiments were conducted with difference equations with the form

\[(1') \begin{cases} \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = f & \text{in } \Omega, \\ u(x, y) = g & \text{on } \partial \Omega. \end{cases} \]

The finite-difference equivalent to the partial differential equation is

\[(1') \begin{cases} \frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = f & \text{in } \Omega, \\ \eta(x, y) = 0 & \text{on } \partial \Omega. \end{cases} \]

The finite-difference equivalent to the partial differential equation is

\[(1') \begin{cases} \frac{\partial^2 \eta}{\partial x^2} + \frac{\partial^2 \eta}{\partial y^2} = f & \text{in } \Omega, \\ \eta(x, y) = 0 & \text{on } \partial \Omega. \end{cases} \]

The examples, for instance, the standard example, where $u$ is the space of the procedure

\[ \eta(x, y) = (1-x)(1-y) \]

and the coarse-grid correction $f''_i$ is the space of the procedure

\[ f''_i = (1-x)(1-y) \]

The examples of residual fine-to-coarse weights

\[ f''_i = (1-x)(1-y) \]

and the coarse-grid correction $f''_i$ is the space of the procedure

\[ f''_i = (1-x)(1-y) \]

The examples of residual fine-to-coarse weights

\[ f''_i = (1-x)(1-y) \]
The stopping criterion (A.15) is based on the assumption that error components with \( \|I\|_2 / \|\tilde{I}\|_2 \) dominate the process. In the first few CCC cycles, however, lower components are dominant, and the main consideration is to converge them. Hence, at that initial stage, the first CCC criterion should be applied.

The key idea of the multi-level adaptive solutions (Section 4.2) is that the residuals in the \( G_1' \) problem are used to control the residuals in the \( G_1 \) problem. This is achieved by ensuring that the norms of the residuals in the \( G_1' \) problem and the corresponding norms in the \( G_1 \) problem are comparable. They should be comparable to the same continuous norms. Also, if the residuals in the \( G_1' \) problem are the "dynamic" residuals (i.e., computed incidentally to the last \( G_1' \) relaxation sweep, using at each point the latest available values of the relaxed solution), then they should be the \( G_k \) dynamic residuals, unlike the residuals transferred to \( G_1 \) (to define \( f_k \); cf. Section A.4) which must be "static" residuals (i.e., computed over the grid without changing the solution at the same time). If, however, and are static and dynamic, respectively, the parameter \( \eta \) in (A.15) should be multiplied by a certain factor \( j \) (see Section 4.6.2 in [3]).

The stopping criterion of the CCC cycle is

\[
\|I_{G_1'}\|_2 < \eta \|I_{G_1}\|_2
\]

where the iteration is deemed complete by a decision criterion.

\[
\|I_{G_1'}\|_2 < \eta \|I_{G_1}\|_2
\]

This criterion is based on the assumption that the error components with \( \|I\|_2 / \|\tilde{I}\|_2 \) dominate the process. In the first few CCC cycles, however, lower components are dominant, and the main consideration is to converge them. Hence, at that initial stage, the first CCC criterion should be applied.

The key idea of the multi-level adaptive solutions (Section 4.2) is that the residuals in the \( G_1' \) problem are used to control the residuals in the \( G_1 \) problem. This is achieved by ensuring that the norms of the residuals in the \( G_1' \) problem and the corresponding norms in the \( G_1 \) problem are comparable. They should be comparable to the same continuous norms. Also, if the residuals in the \( G_1' \) problem are the "dynamic" residuals (i.e., computed incidentally to the last \( G_1' \) relaxation sweep, using at each point the latest available values of the relaxed solution), then they should be the \( G_k \) dynamic residuals, unlike the residuals transferred to \( G_1 \) (to define \( f_k \); cf. Section A.4) which must be "static" residuals (i.e., computed over the grid without changing the solution at the same time). If, however, and are static and dynamic, respectively, the parameter \( \eta \) in (A.15) should be multiplied by a certain factor \( j \) (see Section 4.6.2 in [3]).
may be lowered by orders of magnitude! without large changes in the efficiency. For example:

For the 5-point Poisson equation with Gauss-Seidel relaxation, injection and linear interpolations, (A.15) yields $y_1 = 0.219$. Numericalexperiment (e.g., with the program in Appendix B) shows that with any $0.001$, the computational work is no more than $25\%$ above the work with $0.0002$, and no more than $100\%$ extra work for any $0.0001$.

A.8. Convergence on the Finest Grid. On the finest grid $GM$ the solution is usually considered converged when the (static) residuals are of the order of the truncation error, in some appropriate norm. One way to estimate the truncation error is to measure the mon coarser grids by (5.7), and extrapolate (taking into account that they are $O(h^9)$). Another, related but more straightforward criterion is to detect when the $GM$ solution has contributed most of its correction to the solution. In the FAS algorithm then the natural place to check is when anew $M_i$ is computed, the convergence test being

\[ |G(i)|_F / |G(i)|_F < \epsilon, \]

where the norms are taken over $G(i)$ (or recursively, over $G(i)+1$, $G(i)+2$, etc.).

A.9. Partial Relaxation Sweeps. A partial relaxation sweep over $G_k$ is a relaxation sweep that may skip some subdomains of $G_k$. (Unlike "selective" relaxation sweeps, which in principle pass through all the grid points, although corrections may not be introduced in some of them. Cf. Section 3.2. A partial sweep may be selective, too.)

Partial sweeps are not used much in standard relaxation calculations. Usually, a slow-to-converge subdomain is coupled to others subdomains and therefore cannot be relaxed separately. In the multi-grid process, however, only high-frequency error components are to be reduced by relaxation, and this can be done separately in subdomains: With regard to high frequencies, subdomains are practically decoupled. Hence, in the multi-grid process, partial sweeps are potentially very important. In fact, high-frequency amplitudes may vary greatly over the domain, especially if $\phi$ and $\lambda$ vary much, or if high-frequency error components are introduced at boundaries, making partial sweeping there very desirable.

Partial sweeping may be performed by applying a criterion for slow convergence (Section A.6) separately in subdomains. A subdomain may be excluded from subsequent relaxation sweeps if slow convergence is shown simultaneously on that subdomain and on all neighboring subdomains. Under-relaxation may be used to phase-out the relaxed region (cf. (37), Section 4.6.4). The subdomains may be chosen quite arbitrarily, but each of them should be large enough (at least $4 \times 4$) to allow for separate smoothing.

The other modification is in (A.17), where it was assumed that $GM$ is the finest level everywhere. Generally, the convergence test can be, for example,

\[ \|G(i)|_F / \|G(i)|_F < \epsilon, \]

where the norms are taken over $G(i)$ (or recursively, over $G(i)+1$, $G(i)+2$, etc.).

A.10. Convergence Criteria on Nonuniform Grids. When $G_k$ and $G''$ are not extensive (i.e., the domain covered by $G_k$ is only part of the $G_k'$ domain; cf. Section 7.2), the convergence criteria (Sections A.1—A.8) should be slightly modified. In the multi-grid algorithm (Section 7.2), the convergence criterion (Section A.7) should be slightly modified. The coarsest grid (the domain covered by $G_i$) is only part of the $G_{i-1}$ domain, so the

\[ \|u(i)|_F / \|u(i)|_F < \epsilon, \]

injection and linear extrapolation must be more straightforward criterion in this case. Then when the algorithm is used to produce the final solution, the coarsest grid must be given extra work (cf. (27), Section 7.2).
routine. This essentially reduces the programming of any multigrid solution to the programming of a usual relaxation routine, see Section 3.3. The only subroutines to be changed are the relaxation routine RELAJ and the residual injection routine RESCAL. For different domains, more general GRDFN and KEY subroutines should be written. A general GRDFN subroutine, in which the domain characteristic function is written, is available.

For different domains, more general GRDFN and KEY subroutines should be written. A general GRDFN subroutine, in which the domain characteristic function is written, is available.
Choosing \( r = 1 \) and \( s = 107 \), \( \sigma_0 = 1.55 \) (to minimize \( q \)), straightforward algebraic calculations show that \( q < 0.15 \), and \( 7 < 1.8 \). From (C.5) it follows.

### Appendix B

The Gauss-Seidel sweep employed there can be done in 5 operations per point. But for every 3 sweeps on the grid, interpolations 1 and 2 are also performed, respectively costing 6 and 10 operations per grid point. Hence, a work unit in Appendix B should be considered as representing \( 3 \times 5 + 6 + 10 = 51 \) operations per point. (These operations involve only additions and shifts.)

### Appendix D

Remarks on Initial-Value Problems

The Multi-Level Adaptive Techniques can be used in time-dependent problems. An obvious use could be made for implicit difference approximations, where a multi-grid algorithm (as in Sections 4 and 5) could be employed to solve at each time step the equations for the new time values. Usually, however, alternating-direction implicit (ADI) discretizations, when available, offer the same stability and greater efficiency. The multi-level approach can be used in cases where the time step size is too small for the full-grid method to be applied. The method may be used in cases where the solution is advanced in time explicitly or by ADI or implicit MG techniques can be used in time-dependent problems. An obvious use could be made for implicit difference approximations, where a multi-grid algorithm (as in Sections 4 and 5) could be employed to solve at each time step the equations for the new time values. Usually, however, alternating-direction implicit (ADI) discretizations, when available, offer the same stability and greater efficiency. The multi-level approach can be used in cases where the time step size is too small for the full-grid method to be applied. The method may be used in cases where the solution is advanced in time explicitly or by ADI or implicit MG methods.
Primal Hybrid Finite Element Methods

For 2nd Order Elliptic Equations

PcneL 1977, Pages 401-413

Mathematics of Computation, Volume 31, Number 14

ACI'H BRANCD