VOL. 18, NO. 10, OCTOBER 1980 AIAA JOURNAL 1165
ARTICLE NO. 79-1455R

Multilevel Adaptive Computations in Fluid Dynamics

Achi Brandt
The Weizmann Institute of Science, Rehovot, Israel

I. Introduction

THE usual approach to solving partial differential boundary-value problems is first to discretize the problem in some preassigned manner (e.g., finite element or finite difference equations on a fixed grid), and then to submit the resulting discrete system to some numerical solver. In the multilevel adaptive technique (MLAT),[1-23] discretization and solution processes are intermixed. A sequence of uniform grids (or "levels"), with geometrically decreasing mesh sizes, participates in the process. The cooperative solution process on these grids involves relaxation sweeps over each of them, coarse-grid-to-fine-grid interpolations of corrections and fine-to-coarse transfers of residuals. This process has several important benefits. First, it acts as a very fast solver of the algebraic system of equations, since relaxation on each level is very efficient in liquidating those error components whose wavelength is comparable to that level's mesh size. General nonlinear boundary-value problems, such as Navier-Stokes equations in a general domain, are solved at computational work comparable to seven or so relaxation sweeps on the finest grid. The computer storage used may be much smaller than the number of discrete unknowns. Sections II-IX of this paper survey these fast solvers, emphasizing recent work[8,10,11] on elliptic systems, such as Cauchy-Riemann, Stokes, and Navier-Stokes equations. A new type of relaxation, called distributive Gauss Seidel (DGS), has been developed for such systems, based on the elliptic decomposition of the "symbol" of the finite-difference operators.[8] Also mentioned are fast solvers for compressible viscous flows, for transonic problems, and for unstable steady-state flows.

Moreover, the fast multilevel solutions can actually represent better approximations than the full solutions of the difference equations. For example, one can combine the stability of upstream differencing with the accuracy of central differencing by using the first in relaxation and the latter in the residual transfers. One can obtain extrapolations to higher-order approximations by a trivial change in a lower-order program. Or one can solve evolution problems very inexpensively by performing most time steps on coarse levels, in such a way that the finest-level accuracy is still maintained. These and related techniques (for ill-posed problems, bifurcation problems, and for parametric optimization) are surveyed in Sec. X.

Finally, the multilevel structure provides, in a natural way, very flexible and adaptive discretization schemes, which can automatically and efficiently treat boundary layers and other singularities[7] (Sec. XI).

This is a survey paper. Naturally, not all the ideas expressed here can be rigorously supported at this time. They are described in more detail elsewhere.[2-8]

II. Basic Processes

To understand the basic numerical processes of MLAT, consider first the usual situation where a partial differential problem

$$LU(x)=F(x), \quad \text{for } x=(x_1,\ldots,x_d) \text{ in a domain } \Omega\subset\Re^d \tag{1}$$

is discretized in a preassigned manner on a given uniform grid G^h, with mesh size h, yielding the finite-difference equations

$$L^hU^h(x^h)=F^h(x^h), \quad (x^h\in G^h) \tag{2}$$

Here $U=(U_1,U_2,\ldots,U_q)$ and its discrete approximation U^h are q-dimensional vectors of unknown functions, L is a linear or nonlinear differential operator and $L^hU^h(x^h)$ is, correspondingly, a linear or nonlinear expression involving values of U^h and x^h at neighboring grid points. At various instances of the solution process we have on G^h an approximation to U^h, which we will generally denote by u^h.

To obtain a fast solution to Eq. (2) via the multigrid method, we add to G^h a sequence of coarser uniform grids. Let G^H be such a coarser grid; e.g., let the grid lines of G^H be every other grid line of G^h, so that its mesh size is $H=2h$.

One way of inexpensively obtaining an approximate solution u^h to Eq. (2) is to first obtain an (approximate) solution u^H to the corresponding coarser problem

$$L^HU^H(x^H)=F^H(x^H), \quad (x^H\in G^H) \tag{3}$$

Achi Brandt received his B.S. in Mathematics and Physics from the Hebrew University in Jerusalem in 1960. Following his service in the Israeli Army, he returned there in 1963 to get his M.Sc. He received a Ph.D. from the Weizmann Institute of Science in 1965 for his thesis on numerical methods in hydrodynamics and magnetohydrodynamics. Over the next 10 years, Brandt served in positions of ever-increasing responsibility at that Institute, including Head of the Department of Pure Mathematics from 1973 to 1975. During that period, he also spent a year at the Courant Institute of Mathematical Sciences of New York University and a year at the Departments of Mathematics and Computer Sciences of Stanford University. In 1975 and 1976 Brandt was Senior Visiting Scientist in the Department of Mathematics at IBM's Watson Research Center and in 1977 was Visiting Scientist at the Institute of Computer Applications in Science and Engineering at NASA's Langley Research Center. In 1978, Achi Brandt returned once more to the Weizmann Institute, where he is now Professor and Head of the Department of Applied Mathematics. He is a member of the Israel Mathematics Society and the Society for Industrial and Applied Mathematics (USA).

Presented as Paper 79-1455 at the AIAA 4th Computational Fluid Dynamics Conference, Williamsburg, Va., July 24-26, 1979; submitted Aug. 15, 1979; revision received March 12, 1980.
Index categories: Computational Methods; Subsonic Flow; Transonic Flow.

(which is much less expensive to solve since it contains far fewer unknowns) and then to interpolate u^H to the fine grid:

$$u^h = I_H^h u^H \tag{4}$$

The symbol I_H^h stands for the operation of interpolating from G^H to G^h.

How good the approximation (4) is depends on the smoothness of the solution U^h. In some cases U^h is so smooth that, if the interpolation I_H^h and the coarse grid operator L^H are of order high enough to exploit that smoothness, then u^h obtained by Eq. (4) satisfies

$$\|u^h - U^h\| \leq \|U - U^h\| \tag{5}$$

in some suitable norm. This means that u^h solves Eq. (2) "to the level of the truncation error," which is all we can meaningfully require. In such cases, however, the fine grid G^h is not really needed: the coarser grid G^H already yields a solution with the required accuracy. If G^h is at all needed, our first approximation Eq. (4) will require a considerable improvement.

Can we compute a correction to u^h again by some interpolation from the coarse grid G^H? Namely, can we somehow approximate the error $V^h = U^h - u^h$ by some V^H computed on G^H? Normally, the answer is no. If u^H in Eq. (4) is a good enough approximation to U^H, then V^h will be a rapidly oscillating function that is not visible on the coarse grid G^H. Therefore, before we can reuse the coarse grids, the error V^h should be smoothed out.

An efficient smoothing is obtained by relaxation sweeps. A standard example is the Gauss-Seidel relaxation sweep. This is a process in which all points x^h of G^h are scanned one by one in some prescribed order. At each point the old value $u^h(x^h)$ is replaced by a new value, which is computed so that Eq. (2) is satisfied at that particular point x^h [or nearly satisfied, in case Eq. (2) is nonlinear at that point; one Newton step of changing $u^h(x^h)$ is enough]. Having completed such a sweep, the system Eq. (2) is not yet solved, because its equations are coupled to each other; but the new approximation u^h is hopefully "better" than the old one.

In fact, a well-known, and extensively used, method for solving Eq. (2) is by a long sequence of relaxation sweeps. The trouble is the slowness with which such a sequence converges. The larger the grid, the slower the convergence. The reason is the local nature of the relaxation process. It solves the equations locally, with only slight global effect. In other words, relaxation can have only a small impact (per sweep) on smooth errors, since such errors exhibit only small local errors (small residuals) compared with their own magnitude. On the other hand, relaxation sweeps can be very efficient in performing the local task of smoothing the error. In just a few sweeps of a suitable relaxation scheme, the high-frequency components of the error (i.e., components with wavelength comparable to h) are reduced by an order of magnitude.[3]

Thus, after a couple of relaxation sweeps, the error V^h is smooth. Hence a good approximation V^H to it can inexpensively be computed on the coarser grid G^H. For this purpose, the fine-grid equation

$$L^h(u^h + V^h) - L^h u^h = r^h \tag{6}$$

where

$$r^h \equiv F^h - L^h u^h \tag{7}$$

[equivalent to Eq. (2) but in the form sensitive to V^h] is approximated by the coarse grid equation

$$L^H(I_h^H u^h + V^H) - L^H(I_h^H u^h) = \bar{I}_h^H r^h \tag{8}$$

where I_h^H and $\bar{I}_h^H$ are interpolation operators (not necessarily the same) from G^h to G^H. To avoid complicated linearization in solving Eq. (8), a new unknown function $\bar{U}^H = I_h^H u^h + V^H$ is introduced (instead of V^H) on G^H, in terms of which Eq. (8) becomes

$$L^H \bar{U}^H(x^H) = f^H(x^H), \quad (x^H \in G^H) \tag{9}$$

where

$$f^H = L^H(I_h^H u^h) + \bar{I}_h^H r^h \tag{10}$$

The advantage of Eq. (9) is that it has the same form as Eqs. (2) and (3). Hence, the same routines (e.g., the same relaxation routine), can be used in treating all of them.

The new unknown $\bar{U}^H$ represents, on the coarse grid, the sum of the basic approximation u^h and its correction V^h. This scheme is therefore called the full approximation scheme (FAS)—to be distinguished from the correction scheme (CS), which directly uses V^H. The correction scheme is messy in nonlinear problems, and cannot be applied on composite grids.[3]

Once Eq. (9) is solved (or approximately solved), its solution is used to correct the basic approximation u^h. In doing so we should keep in mind that $V^H = \bar{U}^H - I_h^H u^h$, and not $\bar{U}^H$ itself, is the function that approximates a smooth fine-grid function. Thus, denoting our approximate solution to Eq. (9) by u^H, the corrected approximation on the fine grid should be

$$u^h_{\text{new}} = u^h_{\text{old}} + I_H^h(u^H - I_h^H u^h_{\text{old}}) \tag{11}$$

To summarize the basic processes just described, an initial approximation (4) is obtained from an approximate solution u^H to the coarse grid equation (3) to solve the fine-grid equation (2). Then the approximation is improved by a "multigrid cycle." This cycle includes a couple of relaxation sweeps followed by the "coarse-grid corrector" Eq. (11), in which u^H is an approximate solution to the coarse-grid correction Eqs. (9).

In most cases, at the end of the multigrid cycle the approximation u^H will satisfy Eq. (5) and therefore requires no further improvement.

If for any reason a greater accuracy in solving Eq. (2) is desired, additional multigrid cycles can be performed. Typically, each cycle which includes three sweeps of a suitable relaxation scheme will reduce $\|u^h - U^h\|$ by a factor of 0.2-0.08.

We still have to specify how the coarse-grid equations, first Eq. (3) and later Eq. (9), are actually solved. They are solved in the same way that Eq. (2) is solved, namely, by a combination of relaxation sweeps and coarse-grid corrections, using a grid still coarser than G^H.

III. Multigrid Algorithms

The full algorithm has several variations. One is flowcharted here as Fig. 1. This is a fixed algorithm, its entire flow being prescribed in advance. The particular algorithm shown can be used for solving (to errors below truncation errors) any problem with relaxation smoothing factor (see Sec. IV), uniformly less than $2^{-p/3}$, where p is the approximation order. Most problems (e.g., Poisson, Cauchy-Riemann and Stokes equations and Navier-Stokes equations for moderate R) fall into this category. In other problems one may have to use accommodative algorithms, in which switching between levels is determined by internal checks. For example, the switch back to a coarser grid is made whenever the relaxation exhibits convergence rate slower than the theoretical smoothing rate. Such an algorithm is described in Ref. 4 and flowcharted in Refs. 7 and 8.

In contrast to these full multigrid (FMG) algorithms, cycling algorithms are simplified ones that start with any first approximation on the finest grid, and just perform a (larger) sequence of multigrid cycles. Examples are cycles A, B, and C.[2,3] Cycle C is used in the first stages of developing multigrid codes, since it is the simplest to write and debug. Its asymp-

totic convergence factor can always be compared with the theoretically predicted factor (see Sec. IV).

The typical efficiency of multigrid solvers is clearly seen in the fixed algorithm of Fig. 1. If we take as our work unit the work of one relaxation sweep on the finest level M, it is clear that a sweep over level k costs roughly $2^{-d(M-k)}$ units, and all the relaxation work in the algorithm accumulates to about

$$3(1\cdot 1+2\cdot 2^{-d}+3.2^{-3d}+\ldots)=3/(1-2^{-d})^2 \text{ work units} \tag{12}$$

The other work (mostly interpolations and residual transfers) is quite small compared to Eq. (12) so the overall work should not exceed 7-9 work units. In case of the 5-point Poisson problem ($d=2$), for example, the solution requires less than $42n_M$ operations, where n_M is the number of unknowns at the finest level M, and the operations are, in effect, just additions. The efficiency is independent of the shape of the domain, the type of the boundary conditions, the smoothness of the solution (for improvements in case of smoothness, see Sec. X), and the mesh size. Essentially the same number of work units is required even if the problem is nonlinear (see Sec. X, concerning continuation, or embedding, processes), except when severe instabilities are encountered (cf. Sec. VIII).

The storage requirement of multigrid solvers is minimal. Certain versions can operate with a storage that is even much smaller than n_M, without using an external memory (see Sec. X).

Parallel and vector processing can be fully exploited by multigrid algorithms, since all the expensive operations (interpolations, residual transfers, and suitable versions of relaxation) can be done simultaneously at all grid points. The solution of 5-point Poisson problems, for example, would require 42 $(n/P)+O(\log^2 n)$ steps on a computer with P parallel processors. The running time may well be dominated by set-up times.[21]

IV. Theoretical Predictions and Implementation Aspects

An important feature of multigrid solvers is that the computational work needed to solve a problem is fully predictable through the local mode analysis. This is a simple analysis that can be applied for general nonlinear problems in the following way: The difference equations are linearized around a solution, and the coefficients of the linearized equations are frozen at their local value (or at some local mode of oscillation). The resulting constant-coefficients (or simple-oscillations-coefficients) problem is then assumed to hold in a grid covering the entire space, and its convergence properties can be calculated in terms of the Fourier components of the error. This local analysis fails to approximate the behavior of the long Fourier components, that interact at long distances and are therefore influenced by boundaries and by variations in the coefficients. But these long components may be ignored in the multigrid work estimates, since their convergence is obtained on coarser grids, where the computational work is negligible. The expensive process is that of smoothing the error on the finer grids, and this is essentially a local process that is very well approximated by the local mode analysis.

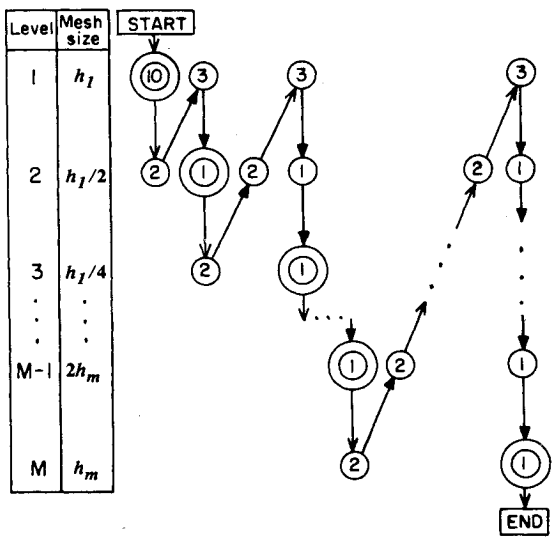

Fig. 1 Fixed full multi-grid algorithm. The number of relaxation sweeps (on the level corresponding to that line) performed at that stage is shown circled. A double circle indicates the stage where the error on that level is guaranteed to be less than the truncation error. The transfer ↓ is the first-approximation interpolation Eq. (4). The transfer ↗ back to the coarse grid introduces there Eq. (9) as well as the first approximation $u^H=I_h^H u^h$. Where the modified arrow ↗ is shown, Eq. (9) can be modified to include τ extrapolation (see Sec. X). The transfer ↓ is the correction interpolation Eq. (11).

Indeed, the predictions of this analysis turn out to be very precise; so much so that they can be used in developing the programs. As long as the cycle C test algorithm does not attain the convergence factor (per work unit) calculated by the local mode analysis, it must contain a programming bug or a conceptual error. Such errors are very common with inexperienced multigridders (especially in their treatment of boundary conditions), hence, it is highly recommended that codes be gradually developed from simple cases, insisting at each stage on attaining the theoretical convergence factors.

The theoretical analysis, itself fully computerized, can be used for selecting the multilevel parameters, such as the orders of various interpolations, the switching parameters in accommodative algorithms, and the mode of relaxation (pointwise or linewise, marching directions, distributions, etc.).

Relaxation, in fact, is the only part of the algorithm that is very problem dependent, and requires expertise and physical insight. All other parts can, in principle, be programmed once for many (perhaps most) problems. A general software is being developed for this purpose.[4,5,13,16]

A simple example of a (cycle C) multigrid program, exhibiting the typical data structure as well as a typical output, is shown in Ref. 3, and can be used as a model for beginners.

The particular knowledge required for designing good relaxation schemes is similar to the specific knowledge required in discretizing the differential equations, and should similarly be learnt gradually, starting from simplest models. The design of relaxation is much facilitated, however, by a standard gage we have for a priori measuring its efficiency. This gage is the local relaxation smoothing factor $\bar{\mu}$ calculated by the mode analysis. $\bar{\mu}$ is defined as the worst (i.e., largest) factor by which high-frequency Fourier components are reduced in one relaxation sweep. The efficiency of the entire algorithm depends on $\bar{\mu}$. For the Gauss-Seidel relaxation of the standard 5-point Poisson equation $\bar{\mu}=0.5$. The calculations can in many cases be carried out by hand, but for complicated schemes $\bar{\mu}$ is calculated by the computer routine SMORATE, available on MUGTAPE.[13]

V. Cauchy-Riemann Equations

Recently,[8,10,11] multigrid solvers have been developed and tested for various flow problems that constitute elliptic systems. The ellipticity of the differential system, the ellipticity of the difference approximation, and the mode of relaxation are three closely related aspects that should be understood together.[8] Here we discuss some of the main points through the principal examples.

The simplest elliptic system is the nonhomogeneous Cauchy-Riemann pair of equations

$$\partial_1 U_1+\partial_2 U_2=F_1(x_1,x_2) \tag{13a}$$

$$\partial_2 U_1 - \partial_1 U_2 = F_2(x_1,x_2) \tag{13b}$$

where $U_i = U_i(x_1,x_2)$ are the unknown functions, F_i given functions, and $\partial_i = \partial/\partial x_i$, $(i=1,2)$. In matrix-operator form,

$$LU \equiv \begin{pmatrix} \partial_1 & \partial_2 \\ \partial_2 & -\partial_1 \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \end{pmatrix} = \begin{pmatrix} F_1 \\ F_2 \end{pmatrix} \tag{14}$$

The determinant of L is the Laplace operator

$$\det L = -\Delta = -\partial_1^2 - \partial_2^2$$

This means that Eqs. (13), like the Laplace equation, are second-order elliptic, and hence their solution is determined by one condition along the boundary.

Discretization

Approximating Eqs. (13) by central differences on a regular grid would not be elliptic. It would admit periodic solutions, such as

$$U_i^h = \sum_j A_{ij}\cos(\pi x_j/h) + B_i\cos(\pi(x_1+x_2)/h)$$

to the homogeneous equations ($F_1 \equiv F_2 \equiv 0$), and hence would be unstable for such periodic perturbations. The easy way to construct stable (elliptic) approximation to L, is to construct an L^h that approximates L and such that $\det L^h$ is an elliptic approximation to the Laplace operator. Such an approximation to Eqs. (13) is

$$L^h = \begin{pmatrix} \partial_1^h & \partial_2^h \\ \partial_1^h & -\partial_1^h \end{pmatrix}, \quad \text{i.e.,} \quad \begin{matrix} \partial_1^h U_1^h + \partial_2^h U_2^h = F_1^h & (15a) \\ \partial_2^h U_1^h - \partial_1^h U_2^h = F_2^h & (15b) \end{matrix}$$

where

$$\partial_1^h \varphi(x) = (\varphi(x_1+h/2,x_2) - \varphi(x_1-h/2,x_2))/h$$

and similarly ∂_2^h. It is clear that Eqs. (15) are an $O(h^2)$ approximation to Eqs. (13) and that $\det L^h = -\Delta^h = -(\partial_1^h)^2 - (\partial_2^h)^2$ is the usual 5-point Laplacian, which is a good (elliptic) operator. Indeed, denoting $\theta \cdot x = \theta_1 x_1 + \theta_2 x_2$, the "symbol" of Δ^h *is*

$$h^2 \exp(-i\theta\cdot x/h)\ \Delta^h\ \exp(i\theta\cdot x/h) = 4 - 2\cos\theta_1 - 2\cos\theta_2 \tag{16}$$

which vanishes only for the constant case $[\theta = 0 (\mathrm{mod}\ 2\pi)]$. Hence nonconstant periodic functions $u = \Sigma_\theta A_\theta \exp(i\theta\cdot x/h)$ cannot solve the homogeneous equation, so the operator is stable.

The only trouble with Eqs. (15) seems to be the use of function values half-way between grid lines. This trouble is avoided by using a staggered grid (see Fig. 2), where U_1^h and U_2^h are defined, respectively, at the center of vertical and horizontal links, and where Eq. (15a) is centered at cell centers (F_1) and Eq. (15b) is centered at grid vertices (F_2).

Relaxation

Classical relaxation is based on a one-to-one correspondence between equations and unknowns. The basic relaxation step is to satisfy (or oversatisfy, or undersatisfy) one of the discrete equations by changing the corresponding unknown (or satisfy a group of equations by simultaneously changing the corresponding group of unknowns, as in line relaxation). Such one-to-one correspondence is not always natural, or even possible. In our case, it already is clear in the differential system Eqs. (13) that it would be unnatural to regard Eq. (13a) as corresponding to U_1, say, and Eq. (13b) as corresponding to U_2. None of these equations, in fact, is elliptic by itself. In the difference system Eqs. (15) it would be impossible to have even a natural one-to-one correspondence between pairs of equations and pairs of unknowns, and such pair-wise relaxation schemes will fail to converge even if arbitrarily small under-relaxation parameter are used. We shall therefore use "distributive relaxation," i.e., a relaxation scheme that satisfies each discrete equation in its turn by distributing changes to several unknowns, in a natural manner.

A natural relaxation scheme is shown in Fig. 3. Figure 3a exhibits the four changes $\pm\delta$ in the approximate solution $u = (u_1,u_2)$, which are made in relaxing Eq. (15a) for the shown cell. Similarly Fig. 3b exhibits the relaxation of Eq. (15b) centered at the shown grid vertex. In each case δ is chosen so that the changed variables exactly satisfy the relaxed equation. The main property of this relaxation is that in relaxing Eq. (15a) we keep all the residuals of Eq. (15b) unchanged, and vice versa. It is indeed necessary in each case to take the other equation into account, since neither equation by itself is elliptic, only the system is. The scheme preserves the property of Gauss-Seidel schemes that each discrete equation at its turn is fully satisfied, and is therefore called distributive Gauss-Seidel (DGS). For obvious modifications near the boundary, calculation of the smoothing factor and other details, see Refs. 8 and 21.

The smoothing factor of the above relaxation is $\bar{\mu} = 0.5$, which is, unsurprisingly, the same as for the 5-point Poisson problem. Hence, also the same multigrid efficiency is obtained. The algorithm of Fig. 1 solves the problem within the truncation errors (and possibly much below, if τ extrapolations are employed—see Sec. X) in about $61n$ operations (mostly additions), where n is the number of discrete unknowns, and where the domain and boundary conditions are arbitrary. A faster multigrid method exists[8] which solves Eqs. (15) in about $23.5n$ operations (additions only), but it is the method just described which we can extend to general elliptic systems.

VI. Steady-State Stokes Equations

The next system we consider is the Stokes equations for slow incompressible viscous flows in d dimension

$$\nabla \cdot U = F_0 \tag{17a}$$

$$-\Delta U + \nabla P = F \tag{17b}$$

where $U = (U_1,\ldots,U_d)$ is the velocity, P the pressure, $\nabla = (\partial_1,\ldots,\partial_d)$, $\nabla\cdot U = \Sigma\partial_i U_i$, $\Delta = \partial_1^2 + \ldots + \partial_d^2$, and F_0 and $F = (F_1,\ldots,F_d)$ are given forcing functions. The matrix-

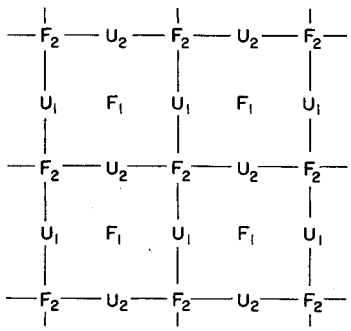

Fig. 2 Discretization of Cauchy-Riemann equations.

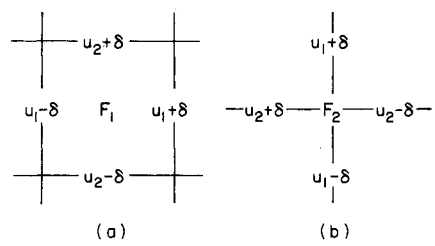

Fig. 3 Relaxation of Cauchy-Riemann equations.

operator form of Eq. (17) is

$$L\begin{bmatrix} P \\ U_1 \\ \vdots \\ U_d \end{bmatrix} \equiv \begin{bmatrix} 0 & \partial_1 \dots & \partial_d \\ \partial_1 & -\Delta & 0 \\ \vdots & & \ddots \\ \partial_d & 0 & -\Delta \end{bmatrix} \begin{bmatrix} P \\ U_1 \\ \vdots \\ U_d \end{bmatrix} = \begin{bmatrix} F_0 \\ F_1 \\ \vdots \\ F_d \end{bmatrix} \tag{18}$$

hence det $L=(-\Delta)^d$, and Eqs. (17) are an elliptic system of order $2d$, requiring d boundary conditions. Usually, these conditions are the values of U on the boundary.

We discretize Eqs. (17) on a staggered grid (Fig. 4) using the $O(h^2)$ difference approximation

$$\sum_{i=1}^{d} \partial_i^h U_i^h = F_0^h \text{ at cell centers} \tag{19a}$$

$$-\Delta^h U_j^h + \partial_j^h P^h = F_j^h \text{ at } j\text{-face centers} \quad (j=1,\dots,d) \tag{19b}$$

where j-faces are cell faces perpendicular to x_j, and $\Delta^h=\Sigma(\partial_i^h)^2$ is the standard discrete Laplacian. Clearly det $L^h=(-\Delta^h)^d$ is a nice elliptic operator.

Distributive Relaxation

The jth momentum Eq. (19b) is elliptic in U_j^h, hence its relaxation at each point x is performed by changing $u_j^h(x)$ (the approximation to U_j^h) only. Having done this at all suitable points x for $j=1,\dots,d$, there remain only the continuity Eq. (19a) to be relaxed and the variable P^h to be changed. They are indeed defined on the same set of points, but P^h does not even appear in Eq. (19a). As in the previous section, however, we note that Eq. (19a) by itself is not elliptic. It is part of an elliptic system, and should therefore be relaxed in a way preserving the residuals of the other equations in the system. Such a DGS relaxation step is shown in Fig. 5, where δ is chosen so that the changed variables satisfy the relaxed equation (the continuity equation at the center). For modifications in the difference equations and their relaxation near boundaries, and for the smoothing factor calculation, see Ref. 8.

The smoothing factor is again the same as for Poisson problems: $\bar{\mu}=0.5$ for $d=2$, and $\bar{\mu}=0.567$ for $d=3$. The multigrid algorithm of Fig. 1 can therefore be used. It solves the difference Eqs. (19) to within truncation errors in about seven work units. This was confirmed by various numerical experiments.[11] We also used a cycling algorithm and found the asymptotic convergence factor to be precisely as predicted by the local mode analysis.[8]

VII. Incompressible Navier-Stokes Equations

The steady-state incompressible Navier-Stokes equations in d dimensions are

$$\nabla \cdot U = F_0 \tag{20a}$$

$$QU + \nabla \cdot P = F \tag{20b}$$

where $Q=-\Delta+R\Sigma U_j\partial_j$, R being the Reynolds number. Equations (17) are the special case $R=0$. In the discussion below (but not in the solution process) we treat Q as being independent of U. This is equivalent to linearizing the system around the current solution and omitting the lowest order term of the linearized equations (a term which is, for all R, locally dominated by Q). Then, similar to Eq. (18), we have det $L=-\Delta Q^{d-1}$, which is again $2d$-order elliptic and therefore requires d boundary conditions (usually the d values of U). The discretization is carried out on the same staggered grid (Fig. 4), using the difference equations

$$\sum_{j=1}^{d} \partial_j^h U_j^h = F_0^h \qquad \text{at cell centers} \tag{21a}$$

$$Q^h U_j^h + \partial_j^h P^h = F_j^h, \quad \text{at } j\text{-face centers } (j=1,..,d) \tag{21b}$$

Hence det $L^h=-\Delta^h(Q^h)^{d-1}$, which is nicely elliptic provided Q^h is a "nicely elliptic" approximation to Q. This is to say that the "symbol" of Q^h, namely $\exp(-i\theta\cdot x/h)Q^h\exp(i\theta\cdot x/h)$, should be nicely (uniformly in R) bounded away from 0, for all θ bounded away from 0.

The $-\Delta$ term in Q is approximated by $-\Delta^h$ as before. More problematic are the terms ∂_j, which cannot be approximated by ∂_j^h, since $\partial_j^h U$ are values not on our grid. The central approximation ∂_j^{2h} can be used instead, but for large values of hR this will no longer be properly elliptic. (When $hR\,|U|=O(1)$, $-\Delta^h$ no longer dominates the local properties of Q^h and the principal part of the discrete system no longer corresponds to the principal part of the differential system).[8] Upstream differencing are therefore usually used when hR is large. The $O(h^p)$ upstream approximation to ∂_j is given by[7]

$$\partial_j^{(p)} = \partial_j^{2h} \sum_{q=0}^{[p/2]} a_q(-\delta_{jj}^h)^q + \frac{1}{2h} a_{[p/2]}(-\delta_{jj}^h)^{[(p+1)/2]} \sigma_j(U_j^h) \tag{22}$$

where

$$a_0=1, \quad a_q=a_{q-1}/(4+2/q), \quad \delta_{jj}^h=(h\partial_j^h)^2$$

and $\sigma_j(v)=\text{sign } v$

for odd p,

$$\sigma_j(v)=h\partial_j^{2h} - \tfrac{1}{2}\delta_{jj}^h \text{ sign}(v)$$

for even p. Note that for $p>2$ these differences use downstream values too, while upstream-only formulas would be nonelliptic. using Eq. (22) is equivalent to adding $O(h^p)$ artificial viscosity. $p=1$ is commonly used. An attractive multilevel possibility (Sec. X) is to use Eq. (22) only for relaxation, while the residual transfer in Eq. (10) uses a central approximation, such as ∂_j^{2h}.

DGS Relaxation

Generalizing the scheme in Sec. VI to any elliptic operator Q^h, relaxation proceeds as follows. The j Eq. (21b) is relaxed by changing values of u_j^h only, in a manner suitable for Q^h. For example, if each component U_i has a constant sign throughout, a pointwise Gauss-Seidel relaxation marching

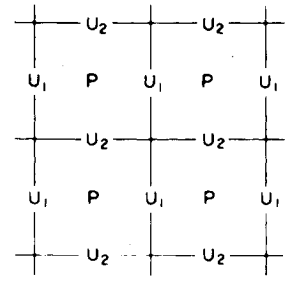

Fig. 4 Discretization of Stokes equations, P^h is defined at cell centers, and U_j^h at j-face centers.

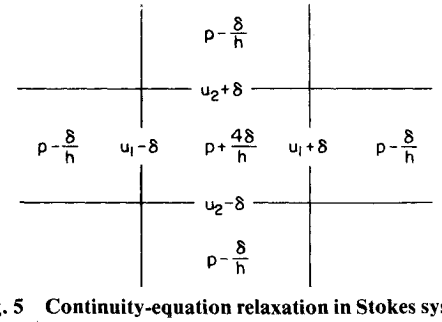

Fig. 5 Continuity-equation relaxation in Stokes system.

downstream is the most efficient manner. If all combinations of signs and relative magnitudes of RhU_i appear, symmetric line relaxation (for $d=2$) or plane relaxation (for $d=3$) may be best. The continuity Eq. (21a) is relaxed in a manner generalizing Fig. 5. Denote by χ_x^h the characteristic function of the cell whose center is at x. This is, x_x^h is defined at cell centers, $\chi_x^h(\xi)=0$ for $\xi \neq x$, $\chi_x^h(x)=1$. Then the relaxation of Eq. (21a) at the point x is described by the changes

$$u_j^h \leftarrow u_j^h - \delta h \partial_j^h \chi_x^h \tag{23a}$$

$$p^h \leftarrow p^h + \delta h Q^h \chi_x^h \tag{23b}$$

where δ is chosen so that the changed u_j^h satisfy Eq. (21a) at x.

It is easy to see that Eqs. (23) are such that the residuals of Eq. (21b) are preserved at least in the approximate sense of regarding Q^h as locally constant. Hence it is easy to show that the smoothing factor of our relaxation scheme (always defined in that approximate sense) is still $\bar{\mu}=0.5$ for $d=2$ and $\bar{\mu}=0.567$ for $d=3$. The multigrid algorithm of Fig. 1 can therefore be used, solving Eq. (21) in about seven work units.

Numerical experiments[11] show that this efficiency is indeed obtained, expect when R becomes comparable to physically unstable values. Improved relaxation is described in Ref. 21.

VIII. Physical Instabilities

The main difference between physical and numerical instabilities is that the latter first appear at high frequencies (where the numerical solution does not approximate the differential one) while the former first appear at low-frequency modes, modes whose Reynolds number (i.e., R times their wavelength) is large. The slow divergence of such smooth modes in a relaxation process (which may be regarded as a time-like process) does not trouble the error-smoothing process. Also, the instability does not appear when the multigrid process (see Fig. 1) first works at coarse levels, since the numerical scheme contains enough artificial viscosity [see Eq. (22)]. It is only when the process comes back to coarse levels after visiting sufficiently fine levels (where artificial viscosity is sufficiently small) that the physical instability starts to show up.

Even for stable Reynolds numbers, but close to unstable ones, the effect is already felt as deterioration in the smoothing and convergence rates of the coarse-grids relaxation [when solving Eq. (9), i.e., the coarse-grid equations as corrected by the fine grids]. Hence it is necessary at large Reynolds numbers to work with accommodative algorithms (see Sec. III). Such algorithms will spend more sweeps and cycles at coarse levels. The overall efficiency, however, will not be affected much, since those extra sweeps and cycles cost very little. This was confirmed by numerical experiments.

It is clear that no purely iterative (time-like) solution process can solve the steady-state flow equations when the solution is unstable. In the multigrid process, however, this limitation is, in principle, removed, since, at each cycle, the coarsest-grid equations can be solved directly, not by relaxation. In this way it is possible to calculate unstable solutions, provided the coarsest grid used is fine enough to resolve the unstable modes.

IX. Compressible Navier-Stokes and Transonic Flows

The scheme of Sec. VIII can be generalized to solve the steady-state compressible Navier-Stokes equations. A relaxation scheme has been devised which still yields the same smoothing factors and thus will probably give a multigrid solution in seven or so work units. This has not yet been programmed.

During 1975-76 multigrid programs were developed[3,19] for the steady-state small-disturbance transonic flow problems, in which the equations are mixed elliptic-hyperbolic, and the solutions contain shocks. Although these programs are obsolete (with the present stage of multigrid experience and knowledge they could be corrected and improved in various ways), they do clearly show that the typical multigrid efficiency can be obtained for this type of problems. The efficiency deteriorated when relaxation schemes were used which have bad theoretical smoothing factors, and/or when the treatment of boundary conditions was too disturbing for the interior smoothness. These flaws can be removed. For more recent developments see Refs. 22 and 23.

X. Multilevel Super Solvers. Evolution Problems

On top of their role as fast solvers of the algebraic systems arising in discretizing continuum problems, the multilevel processes offer other very important advantages. Some of these are surveyed below.

One advantage is related to the effective separation between the high frequencies (affected only be relaxation) and low frequencies (affected mainly by the coarse-grid corrections). This can be used in various ways. For example, a basic issue in computational fluid dynamics is when to use upstream differencing, which is less accurate but more stable than the corresponding central differencing (see, for example, Sec. VII). Upstream differencing is much better for the highest-frequency modes, while central differencing is better for lower modes, and especially for the lowest ones. This conflict may be completely resolved in multilevel processes, where one can use the upstream differences in the relaxation process, but transfer residuals [r^h in Eq. (10)] computed by central differencing. This will ensure stability (and hence also efficient smoothing) together with the higher-order accuracy. Note that the process will not converge to zero residuals since it uses two conflicting difference schemes. The very point is, however, that the solution produced is a better approximation to the differential solution than can be produced by either scheme.

The effective separation can also be used to solve ill-posed problems, where the solution is required to fit some data that are not the normal, well-posed boundary conditions. In such situations only smooth components of the solution can meaningfully be fitted to the data (or smooth averages of data). The multilevel process will therefore include data fitting only at its coarse-level phases. These phases cost very little, hence the solution of such ill-posed problems will cost only slightly more than solving the correspondingly well-posed problems.

We often need to solve not just one isolated problem but a sequence of (many) similar problems depending on some parameter. For example, we may be studying the effect of changing some physical parameters on the "performance" of a system, which we may moreover want to optimize. Sequences of problems are also encountered in continuation (or "embedding") processes for solving nonlinear problems,[3,6] especially in following the paths of bifurcating solutions. Or, most frequently, the sequence of algebraic systems may represent the implicit discretization of an evolution problem, one algebraic system per time step. Such sequences can be handled very efficiently by multilevel processes, by expoilting the very different rates of change of high-frequency and low-frequency components. Usually, the high-frequency ones change faster in the first few steps, but then they settle down. The changes in the solution are then smooth and can accurately be described on increasingly coarser levels. (This, incidentally, allows the use of large time steps without employing implicit equations at all!) Only once in many steps should a step be made again on finer levels, to readjust the high-frequency components to the changes that have developed in the smooth components.

The multilevel technique for freezing high frequencies and working on coarse levels only is simply to use Eq. (9) and freeze the values of $\tau^H = f^H - F^H$, where F^H is the original right-hand side in Eq. (3). τ^H is actually the local truncation error of grid H relative to the finest grid (i.e., τ^H is the

correction to the level-H equations which makes their solution coincide with the finest-level solution). By freezing τ^H at its value in a previous step we work on coarse levels but without ignoring high frequencies; only high-frequency changes are ignored. It is essential of course that when a finer level is revisited after several steps, the interpolation Eq. (11) is used. Observe that this interpolation preserves the high-frequency content of the fine grid.

Employing this technique for parabolic time-dependent problems in particular, we get a process of marching in time which is mostly performed on coarse levels, while the finest-level accuracy is essentially maintained. In some examples, (e.g., the heat equation in an infinite domain) one can prove (by Fourier analysis) that marching in this way from initial state to 90% steady steate costs less than the work of 10 explicit time steps.

The relative local truncation error τ^H can be used in various other ways. Since it approximates the true local truncation error τ_0^H it can be used in switching and grid adaptation criteria (see Sec. XI). Moreover, Taylor expansions (in H) easily give the first term $A_0(x)$ in the expansion of $\tau_0^H(x)/\tau^H(x)$. Multiplying $\tau^H = f^H - F^H$ in Eq. (9) by $A_0(x)$ will provide higher-order accuracy for negligible extra work. This slight change in the multigrid algorithm (see Fig. 1) is called τ extrapolation.[7,10,11] It extrapolates the equations, whereas the more familiar Richardson extrapolation works on the solutions and is therefore restricted to the case that A_0 does not depend on x.

Another interesting possibility in multilevel processing are algorithms that require only small computer memories, without using any external storage and without losing too much in efficiency. They are based on the observation that the finer level is actually needed only for calculating the τ^H correction to the coarser-level equations. The dependence of τ^H on U^H is local, and only a local piece of the finer grid is needed in order to calculate it. The finer level can thus be used one piece at a time, while at subdomains not currently covered by the fine piece frozen values of τ^H are employed. (Cf. segmental refinement[3,4] and the remark at the end of Sec. 2.2 of Ref. 7.)

XI. Nonuniform Adaptive Structures

Perhaps the most important advantage of the multilevel process, in particular for singular perturbation and other irregular problems, is its full compatibility with adaptive processes, where the mesh size, as well as other discretization parameters, can progressively be adapted to the features of the computed solution. In fact, the multilevel process is itself adaptive; its discretization is adapted to the *error* in the solution.

The special capability of the multilevel structure to create very flexible nonuniform discretization patterns is obtained by observing that the various grids (levels) need not all extend over the same domain. Finer levels may be confined to increasingly smaller subdomains, so as to provide higher resolution only where desired. Moreover, we may attach to each of these localized finer grids its own local system of coordinates, to fit curved boundaries or to approximate directions of interior interfaces and thin layers. Unlike global coordinate transformation, these local coordinates do not complicate the difference equations throughout the domain (hence do not turn the one-dimensional trouble of boundary approximations into a two-dimensional trouble of complicated equations). All these patches of local grids interact with each other through the multigrid process, which, at the same time, provides fast solutions to their difference equations (an important advantage over other methods of patching grids or using transformations).

This structure, in which nonuniform discretization is produced through the sequence of uniform grids, is indeed highly flexible. Changing the local mesh size is affected by the relatively simple operation of extending or contracting the domains of uniform grids (an operation completely automated by GRIDPACK[4,12]). Moreover, since in this structure only equidistance differencing is needed (much less expensive than differencing on variable grids), it becomes feasible to employ high-order difference approximations, even in singular-perturbation cases.[7]

The discretization can thus be progressively refined and adapted. The actual adaptive solution process is governed by certain criteria.[3] Derived from optimization considerations, these are local criteria which automatically decide where and how to change the local discretization parameters. The criteria are based on values of the relative truncation errors τ^H (Sec. X), and are controlled by the user through a certain function G (the error-weighting function), which, in effect, expresses the purpose of the numerical calculations; i.e., the sense (or the error norm) in which approximations to the true solution are to be measured.

The resulting discretization will be of high order wherever the evolving solution is suitably smooth. Singularities will automatically be detected and treated, usually by introducing increasing finer levels on increasing smaller neighborhoods of the singularity.

Boundary layers will sometimes be resolved by the adaptive process, sometimes be completely "skipped," depending on the choice of the control function G. In any case, the convergence rate in the suitable sense (i.e., in the error norm corresponding to G) is always fast. The error $E = \|u - U\|$ decreases exponentially as a function of the computational work W, where $u = u(W)$ is the evolving numerical solution, U the true differential solution, and $\|\cdot\|$ the appropriate error norm. That is, the convergence $E \approx C(p)\, W^{-p/d}$ is obtained if fixed p-order differences are used, while $E \approx \exp(-cW^{\alpha})$ is obtained when p is adaptable too. Moreover, these rates are uniform, e.g., in case of the Navier-Stokes Eqs. (20), $C(p)$, c and α are independent of R.[7]

Acknowledgment

The work reported here was partly performed under NASA Contract NAS1-14101 while the author was in residence at ICASE, NASA Langley Research Center, Hampton, Va.

References

[1] Brandt, A., "Multi-Level Adaptive Technique (MLAT) for Fast Numerical Solution to Boundary Value Problems," *Proceedings of the 3rd International Conference on Numerical Methods in Fluid Mechanics*, (Paris, 1972), *Lecture Notes in Physics*, Vol. 18, Springer-Verlag, Berlin, 1973, pp. 82-89.

[2] Brandt, A., "Multi-Level Adaptive Techniques," IBM T.J. Watson Research Center, Yorktown Heights, N.Y., IBM Research Rept. RC6026, 1976.

[3] Brandt, A., "Multi-Level Adaptive Solutions to Boundary-Value Problems," *Mathematics of Computation*, Vol. 31, 1977, pp. 333-390.

[4] Brandt, A., "Multi-Level Adaptive Solutions to Partial Differential Equations—Ideas and Software," *Mathematical Software III, Proceedings of Symposium*, edited by J. Rice, Mathematics Research Center, Univ. of Wisconsin, March 1977, Academic Press, New York, 1977, pp. 277-318.

[5] Brandt, A., *Lecture Notes of the ICASE Workshop on Multi-Grid Methods*, with contributions by J. C. South (Lecture 8), J. Oliger (10), F. Gustavson (13), C. E. Grosch (14), D. J. Jones (15), and T. C. Poling (16), ICASE, NASA Langley Research Center, Hampton, Va., 1978.

[6] Brandt, A., "Multi-Level Adaptive Finite-Element Methods. I. Variational Problems," *Proceedings of Seminar on Special Topics of Applied Mathematics*, edited by J. Frehse, D. Pallaschke and U. Trottenberg, GMD, St. Augustin, W. Germany, Oct. 1979, North-Holland, 1980.

[7] Brandt, A., "Multi-Level Adaptive Techniques (MLAT) for Singular-Perturbation Problems," *Numerical Analysis of Singular Problems*, Proceedings of Conference, Univ. of Nijmegen, The Netherlands, May-June 1978, edited by J.J.H. Miller and P. W. Hemker, Academic Press, London, 1979, pp. 53-142; also ICASE Rept. 78-18.

[8] Brandt, A. and Dinar, N., "Multi-Grid Solutions to Elliptic Flow Problems," *Numerical Methods for Partial Differential Equations,* Proceedings of Advanced Seminar, Math. Research Center, Univ. of Wisconsin, Oct. 1978, edited by S. Parter, Academic Press, New York, 1979, pp. 53-147.

[9] Brandt, A., Dendy, J. E., Jr., and Ruppel, H., "The Multi-Grid Method for Semi-Implicit Hydro-dynamics Codes," LA-UR 78-3066 Rept. of Los Alamos Scientific Laboratory, Los Alamos, N. Mex., 1978; also *Journal of Computational Physics,* Vol. 34, 1980, pp. 348-370.

[10] Dinar, N., "On Serveral Aspects and Applications of the Multi-Grid Method for Solving Partial Differential Equations," NASA Contract Rept. 158947, 1978.

[11] Dinar, N., "Fast Methods for the Numerical Solution of Boundary-Value Problems," Ph.D. Thesis, Weizmann Institute of Science, Rehovot, Israel, 1979.

[12] Hackbusch, W., "On the Convergence of a Multi-Grid Iteration," *Beitr. Numer. Math.,* Vol. 9, 1980.

[13] Brandt, A., Dinar, N., Gustavson, F., and Ophir, D., contributors, "MUGTAPE. A Tape of Multi-Grid Software and Programs," distributed at the ICASE Workshop on Multi-Grid Methods, 1978.

[14] Nicolaides, R. A., "On the ℓ^2 Convergence of an Algorithm for Solving Finite Element Equations," *Mathematics of Computations,* Vol. 31, 1977, pp. 892-906.

[15] Nicolaides, R. A., "On Finite-Element Multi-Grid Algorithms and Their Use," ICASE Rept. 78-8, ICASE, NASA, 1978.

[16] Ophir, D., "Language for Processes of Numerical Solutions to Differential Equations," Ph.D. Thesis, The Weizmann Institute of Science, Rehovot, Israel, 1979.

[17] Poling, T. C., "Numerical Experiments with Multi-Grid Methods," M.A. Thesis, Dept. of Mathematics, The College of William and Mary, Williamsburg, Va., 1978.

[18] Shiftan, Y., "Multi-Grid Method for Solving Elliptic Difference Equations," M.Sc. Thesis (in Hebrew), The Weizmann Institute of Science, Rehovot, Israel, 1972.

[19] South, J. C. Jr. and Brandt, A., "Application of a Multi-Level Grid Method to Transonic Flow Calculations," ICASE Rept. 76-8, NASA, 1976.

[20] Brandt, A., "Numerical Stability and Fast Solutions of Boundary-Value Problems," *Boundary and Interior Layers-Computational and Asymptotic Methods,* edited by J.J.H. Miller, Ed Boole Press, Dublin, 1980.

[21] Brandt, A., "Multigrid Solvers on Parallel Computers," *Proceedings of Conference on Elliptic Problem Solvers* (Santa Fe, June 30-July 2, 1980), edited by M. Schultz, Academic Press, N.Y., 1980.

[22] Jameson, A., "Acceleration of Transonic Potential Flow Calculations on Arbitrary Meshes by the Multiple-Grid Method," AIAA Paper 79-1458, *Proceedings of AIAA Computational Fluid Dynamics Conference,* July 1979, Williamsburg, Va., pp. 122-146.

[23] McCarthy, D. R. and Reyhner, T. A., "A Multi-Grid Code for Three-Dimensional Transonic Potential Flow about Axisymmetric Inlets at Angle of Attack," AIAA Paper 80-1365, *Proceedings of AIAA 13th Fluid and Plasma Dynamics Conference,* July 1980, Snowmass, Colo.