NASA Contractor Report 187595
ICASE Report No. 91-36

ICASE

A DYNAMICALLY ADAPTIVE MULTIGRID ALGORITHM FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS — VALIDATION AND MODEL PROBLEMS

C. P. Thompson
G. K. Leaf
J. Van Rosendale

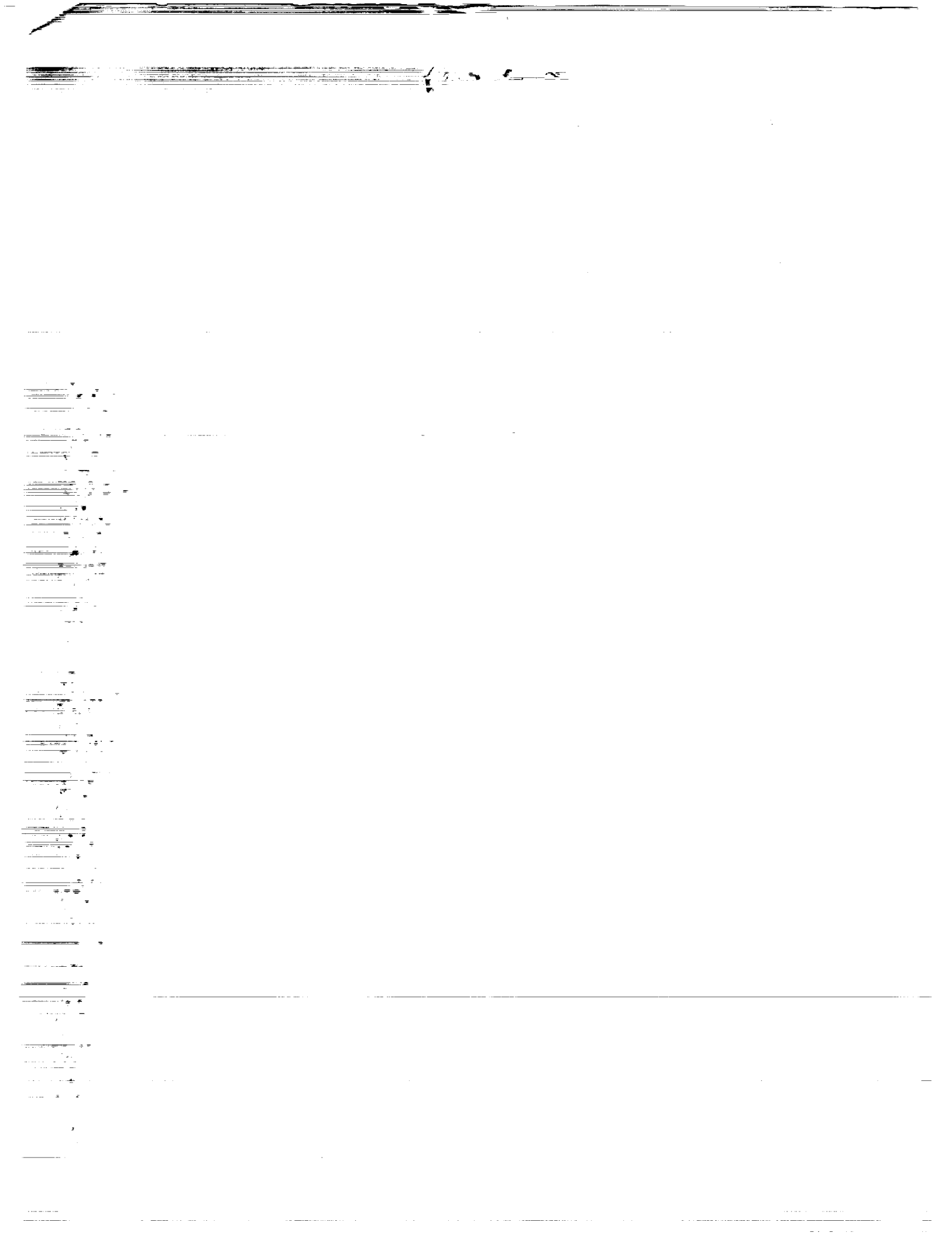
Contract No. NAS1–18605
June 1991

Institute for Computer Applications in Science and Engineering
NASA Langley Research Center
Hampton, Virginia 23665–5225

Operated by the Universities Space Research Association

National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225



A Dynamically Adaptive Multigrid Algorithm for the Incompressible Navier-Stokes Equations -- Validation and Model Problems*

by

C. P. Thompson,** *G. K. Leaf*,† *and J. Van Rosendale*††

Abstract

We describe an algorithm for the solution of the laminar, incompressible Navier-Stokes equations. The basic algorithm is a multigrid method based on a robust, box-based smoothing step. Its most important feature is the incorporation of automatic, dynamic mesh refinement. Using an approximation to the local truncation error to control the refinement, we use a form of domain decomposition to introduce patches of finer grid wherever they are needed to ensure an accurate solution. This refinement strategy is completely local: regions that satisfy our tolerance are unmodified, except when they must be refined to maintain reasonable mesh ratios. This locality has the important consequence that boundary layers and other regions of sharp transition do not "steal" mesh points from surrounding regions of smooth flow, in contrast to moving mesh strategies where such "stealing" is inevitable.

Our algorithm supports generalized simple domains, that is, any domain defined by horizontal and vertical lines. This generality is a natural consequence of our domain decomposition approach. We base our program on a standard staggered-grid formulation of the Navier-Stokes equations for robustness and efficiency. To ensure discrete mass conservation, we have introduced special grid transfer operators at grid interfaces in the multigrid algorithm. While these operators complicate the algorithm somewhat, our approach results in exact mass conservation and rapid convergence.

In this paper, we present results for three model problems: the driven-cavity, a backward-facing step, and a sudden expansion/contraction. Our approach obtains convergence rates that are independent of grid size and that compare favorably with other multigrid algorithms. We also compare the accuracy of our results with other benchmark results.

---

* This research was supported in part by the Applied Mathematical Sciences subprogram of the Office of Energy Research, U.S. Department of Energy, under Contract W-31-109-Eng-38, and in part by the National Aeronautics and Space Administration under NASA Contract No. NAS1-18605.

** Address: Bergen Scientific Centre, IBM, Thormohlensgate 55, 5008 Bergen, Norway.

† Address: Mathematics and Computer Science Division, Argonne National Laboratory, 9700 South Cass Avenue, Argonne, IL 60439.

†† Address: Institute for Computer Applications in Science and Engineering, NASA Langley Research Center, Hampton, VA 23665.

1. Introduction

Computational fluid dynamics is one of several fields where one finds boundary and interior layers in complex interactions. These regions of sharp transition frequently play crucial roles in determining the global solution, despite their small sizes. The need for true local refinement is therefore great.

We use the phrase "true local refinement" to mean the introduction of extra unknowns in areas of interest and only in those areas. This contrasts with approaches that move mesh points into areas of interest but maintain a constant number of unknowns. Our strategy is atypical of finite difference-based codes, including those that use mesh transformations, since arrays are generally the only data structure used. While moving mesh points can improve accuracy, this approach gives limited control over global accuracy. The necessary flexibility is easily provided by finite element codes, but the overhead in maintaining this freedom slows solution speed. The essential issue is one of grid structure and regularity. We enhance a regular mesh with the pointers necessary to allow us to refine specific sections of the grid while leaving the remainder unmodified.

Our approximation is based on the use of finite differences at each grid level. We extend the usual formulation to support locally refined patches. Such an extension requires a data structure that is simple enough to implement efficiently, yet flexible enough to permit the very complicated patterns of local mesh refinement that would be generated by an adaptive mesh refinement strategy. To achieve these goals, we have selected a quad-tree data structure. At any level, the grid is generated by a collection of square grid patches containing N^2 cells, where $N = 2p$ and p is a specified integer greater than one. The collection of patches making up a grid may or may not be contiguous. A grid refinement is generated by selectively refining any or all of the quadrants in any of the patches in the grid. Refinement by quadrants allows finer-grained adaptivity than refining only entire patches. The refined quadrant becomes a patch at the next finer level, with the same structure as its parent patch. Thus implementation is simple, yet the scheme is flexible enough to allow coupling with adaptive mesh strategies. In addition, this quad-tree data structure allows the multigrid algorithm to be easily adapted to shared-memory parallel architectures **[16]**.

Any multigrid algorithm based on the use of nonuniform composite grids must address two numerical issues. First is the issue of approximating the governing equations on a nonuniform composite grid. In particular, we seek consistent approximations that do not violate the conservation laws beyond the intrinsic level of the approximation. Second, the nature of the transfer functions on a composite grid must be specified so that a consistent multigrid approximation results.

Our algorithms were tested on model problems in two dimensions; in this paper we present results on a driven-cavity, the backward-facing step, and a sudden expansion/contraction configuration. These problems are sufficiently demanding to test the numerical procedures, while

being well understood with an abundance of benchmark data. The point is to design algorithms that easily extend to more complex geometries and more difficult flows.

The essential features of our approach are a dynamically adaptive grid composed of different-sized rectangles and a discretization based on hybrid differences, with an interface treatment that conserves mass exactly. The solution algorithm uses a multigrid approach using "boxed-based" relaxation and a variety of cycling strategies. This is similar to the method adopted by Thompson and Ferziger [15]. However, there are a number of important differences. Our adoption of "patches" as our basic data object and our different choice of grid partitioning have allowed us to avoid the step of grouping cells marked for refinement into subrectangles (which is potentially costly [15]). Moreover, we can handle somewhat more general geometries, support more complex refinement patterns, and (as we show in reference [16]) efficiently perform calculations in parallel.

Our experiments have shown that the novel, mass-conserving transfer operators (which are also an integral part of our interface treatment) are both natural to use and accurate. The use of $\left[\tau_h^H\right]$ as an error estimator works well and is also natural in the multigrid context.

2. The Test Problems and Governing Equations

We consider steady, two-dimensional, laminar, incompressible flow. In this section we describe our simplest test problem, namely, flow in a square box that is driven by a moving lid (the so-called driven-cavity problem). In Section 8 we describe the other two geometries we have considered (the backward-facing step and the sudden expansion/contraction) in the context of our results. The code can easily handle any domain defined by horizontal and vertical lines by a natural extension of our technique for handling grid refinement. The problems are well known; we have chosen them because they are geometrically simple but display complex flow structures. Also, there are many well-documented numerical solutions (see, for example, [5], [2], [7], and [17]).

Our governing equations are

$$\begin{aligned}
\frac{\partial u^2}{\partial x} + \frac{\partial uv}{\partial y} &= -\frac{\partial p}{\partial x} + \frac{1}{\mathrm{Re}}\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right], \\
\frac{\partial uv}{\partial x} + \frac{\partial v^2}{\partial y} &= -\frac{\partial p}{\partial y} + \frac{1}{\mathrm{Re}}\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right], \qquad (1) \\
\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} &= 0,
\end{aligned}$$

where the Reynolds number is defined by

$$\mathrm{Re} = \underline{u}\, L/\nu.$$

We take the length scale to be $L = 1$; $\underline{u}$ is the characteristic velocity to be normalized to one. For the driven-cavity problem, the domain is the unit square, and the associated boundary conditions are

$$u = 0\,, \quad v = 0 \quad \text{on the bottom and sides } (y = 0, x = 0 \text{ and } x = 1)$$

and

$$u = 1\,, \quad v = 0 \quad \text{on the top } (y = 1)$$

(see Figure 1(a)). Other problem domains and boundary conditions are shown in Section 8.

Equations (1) represent a set of three coupled nonlinear partial differential equations. The degree of nonlinearity is determined by the Reynolds number (Re). We present two sets of results: one at Re = 1, which is almost linear but allows us to look at certain asymptotic properties of the discretization, and one at Re = 400, which is a reasonably nonlinear problem but still corresponds to stable, steady, physical solutions. Of course, for this test problem, other formulations, such as stream-function vorticity, are available. We have retained the primitive variable formulation because it is easier to generalize to more complex flow regimes and to three-dimensional problems.

In addition to the two corner singularities mentioned, there is the usual pressure indeterminacy, always found with incompressible flows: the pressure is determined only up to an additive constant. Since our algorithm is based on iterative methods, we simply ignore the additive constant. Our iterative scheme converges to a solution with average pressure almost the same as that of the starting guess. We avoid some unnecessary complexity, and we gain in convergence rate by this approach.

3. Basic Discretization

We have chosen a standard staggered grid as the basic discretization scheme. In this formulation the computational domain is filled with "mass control volumes." Pressures are defined in the centers of these rectangles, vertical velocity components are defined at the top and bottom faces, and horizontal ones are similarly offset (see Figure 1(b)). This means that it is not necessary to generate pressure boundary conditions.

We have several reasons for choosing this discretization. Principally, the properties of this approach are well understood, and we wish to avoid possible interactions between newer discretizations and our treatment of the fine/coarse interfaces. The main advantage is the stability of the scheme: it will not exhibit spurious pressure modes, provided only that the momentum equations are stably discretized. Moreover, the discrete ellipticity is somewhat better than in nonstaggered formulations, although these can be corrected by introducing artificial compressibility into the

continuity equation.

Our approximations of derivatives are also quite standard. We treat diffusion terms with standard three-point central differences. The pressure gradient terms are also discretized by using central differences. The convection terms require more careful handling. We use upwind differencing if the mesh Reynolds number (uh/ν) is greater than 2; otherwise, we use central differencing. Thus, for sufficiently small meshes, this discretization has second-order local truncation errors for all variables. An equally important property is that the discretization is stable (h-elliptic) for all flow rates on all grids [13]. This is essential in multigrid methods: violation of this condition can mean that the coarse-grid correction is grossly inaccurate. The actual discrete approximations are given in detail in [1].

Mesh transformations have become an increasingly common technique for adapting finite difference algorithms to complex geometries. Nonstaggered discretizations have an obvious advantage in this situation, since it is necessary to calculate and store only one set of geometric information instead of four (in three dimensions). However, the amount of increased computational complexity in writing the differential equations in transformed coordinates is very large, and it is not clear whether this is the best method for dealing with complex geometries. One approach that fits naturally into our multigrid framework is to have locally fitted grids that overlap with a Cartesian system in the interior (see, for example Chesshire and Henshaw [6]). This method allows efficient treatment of the interiors while accurately representing some classes of complex geometry.

4. A Brief Overview of the Multigrid Solution Algorithm

In the current context, the goal of a good multigrid algorithm can be defined as the robust solution of a set of elliptic equations in an amount of work equivalent to a small number of relaxation sweeps. In particular, we reject strategies where the convergence rates degrade as the meshes become finer. Section 8 describes the specific aims of our project in more detail.

For completeness, we give a brief outline of our multigrid algorithm here. A more complete description of multigrid methods is given in references [4], [9], and [10]; and a description of an algorithm similar to the present one, but for nonadapted grids, is given in [1]. Thus we restrict ourselves here to an overview of our current algorithm, stressing new features where appropriate.

We have already defined the fine-grid representation of our problem. For efficiency, we use geometric coarse-grid operators. (In other words, the coarse-grid operator is defined in the same way as the fine-grid one except that h is replaced by $2h$.)

The staggered discretization works well in the context of multigrid methods and offers a number of benefits. Among the various ways of performing the relaxations, we use the approach

of simultaneously updating all five variables (four velocity components and pressure) associated with a mass control volume. This method seems to be more effective and more reliable than schemes that cyclically update the variables associated with each cell, especially at high Reynolds numbers. At fine/coarse-grid interfaces, the staggered formulation has the advantage that the molecules that must be simultaneously updated are smaller than those in nonstaggered formulations; hence, accuracy may be improved. Details are given later.

Before the update of each cell, we locally linearize about the old solution. There are some indications that we would achieve better rates of convergence if we used local Newton updates. However, the increased amount of arithmetic involved in this process offsets the faster convergence. On the coarsest grid we perform as many sweeps as are necessary to satisfy a convergence criterion. On finer grids we perform a small number of sweeps, as defined by our cycling strategy.

We use full weighting for the multigrid restrictions in all cases. This gives us better convergence properties, since it minimizes aliasing effects. (This technique contrasts with our earlier approach [1], where we used linear interpolation for our restrictions.)

For prolongations of velocities we use an approach that is second order but automatically satisfies the discrete continuity equations on fine cells, provided they were satisfied on the corresponding coarse-grid cell. This property somewhat simplifies our treatment of interfaces. Details are given in the next section. Pressures are prolonged by using linear interpolation. It should be noted that the proof given in [1], showing that discrete continuity on the coarsest grid implies discrete continuity on all grids, is still valid in this context.

We have implemented all the common multigrid cycling strategies (F, W, and V) as well as adaptive cycling. Earlier results have shown that for nonlinear Navier-Stokes equations, F cycles are somewhat more efficient [1]. We have done most of our work with this strategy. However, experiments have shown that the rate of convergence is insensitive to the cycling strategy. Note that our cycling strategy is the same in the uniform and nonuniform grid case: we perform operations on cells of a particular mesh size during each sweep.

5. The Discretization on Compound Grids (Allowable Grids)

It is common, especially in multigrid programs, to use uniform grids at each level of refinement. Such programs are easier to write and avoid the computational overhead of dealing with varying mesh spacing. Of course, in many problems uniform grids lead to excessive problem size in order to achieve a desired degree of resolution in the solution. Our approach is to recover the flexibility of nonuniform grids by generating a composite grid made up of a collection of patches with each patch having a uniform grid. The size of each patch varies according to its level and thereby generates a nonuniform composite grid.

To illustrate this concept, let us start with patches composed of 4 cells. (A patch consists of N^2 cells, where $N = 2p$; in our code we assume that $p > 1$, but we use $p = 1$ in the diagram for clarity.) Consider a square domain (such as a driven cavity) with a composite grid made up of three grid levels. We shall generate a typical composite grid to illustrate the procedure. Suppose grid level 1 consists of one patch that is the entire domain, as shown in Figure 2(a). In this figure we have indicated the positions associated with each of the 6 horizontal velocities, 6 vertical velocities, and 4 pressures defined on a staggered grid associated with a 4-cell patch. Now, each patch is made up of 4 quadrants. As patches for grid level 2, we can select any one, two, three, or all of these quadrants as patches on level 2. For illustrative purposes, we select two patches on grid level 2. Let these patches be the lower left (patch 2) and upper right (patch 3) quadrant of the single patch (patch 1) on level 1 as shown in Figure 2(b). Now consider the third grid level. As candidates for patches on this level, we can select any combination of quadrants in patches 2 and 3. For the sake of simplicity in the figures, suppose we select one patch at level 3 located in the upper right quadrant of patch 3. This is the fourth patch in the composite grid. We note that if we had selected, for example, the lower left quadrant of patch 3 to be a patch on level 3, we would have added four more patches on level 2 so that adjacent patches are separated by at most one level (this includes diagonal neighbors). In Figure 2 we show the composite grid at each level, with the velocity and pressure node locations indicated. Thus, the final composite 3 level grid is made up of 4 patches containing 18 horizontal velocity, 18 vertical velocity, and 13 pressure nodes. Note that, for illustrative purposes, we have used 4-cell patches; in practice we use 16-cell (or larger) patches.

Having illustrated the generation of nonuniform composite grids by means of patches generated by quadrant refinement, we now consider the issues involved in approximating the Navier-Stokes equation on nonuniform grids. In Figure 3(a) we show a typical interface between grids of different sizes. For example, Figure 3(a) could illustrate an interface in Figure 2(b) between a level-1 grid and a level-2 grid, or it could illustrate an interface in Figure 2(c) between a level-2 grid and a level-3 grid.

There are two coarse-grid cells. The right-hand one is subdivided into four fine-grid cells. The staggering of the variables is shown in this diagram. We have shown the locations of all genuine variables in Figure 3(a). (By "genuine" we mean unknowns that are associated with an approximation to some differential equation.) We write a horizontal momentum equation for each u-velocity (the vertical component is treated similarly); continuity equations are written at the centers of cells where the pressure is defined.

The usual five-point molecules that are used away from interfaces must be modified here. For example, the continuity equation for the left-hand cell involves three u-components; the momentum equations are also modified to involve both coarse- and fine-grid quantities. Since this approach is rather inconvenient, we introduce a border of virtual cells in the coarse-grid region (see Figure 3(a)). These are of size h (the fine-grid spacing). In this case seven extra unknowns

are associated with the fine grid. Additionally, four extra unknowns are associated with the coarse grid. Both these sets of unknowns are introduced as a computational convenience to "regularize" the structure of unknowns. Effectively, the new fine-grid cells act as boundary values for the fine-grid equations, and those for the coarse grid have a similar role. However, we have to introduce new matching conditions that are consistent with the differential equations but that are not necessarily derived from them directly. Obviously, some extra work is involved in this process, but in our multigrid context this work is small and can be easily incorporated into our normal transfer operators.

For efficiency reasons we choose to relax only those cells that have a particular mesh size (i.e., at a given grid level) at any one sweep. We relax over the whole domain only on those coarse grids that cover the whole region. Where we specify refined subdomains, we apply the smoother only to that subgrid. This approach follows the multilevel adaptive technique of Bai and Brandt [11] and the fully adaptive composite approach of McCormick [12].

We have constructed this arrangement to allow a "natural" treatment of all the genuine variables. This means that we write the usual discretized partial differential equations for all the variables shown in Figure 3(a). All of these variables are updated by our usual relaxation technique throughout the smoothing phase of the multigrid cycle for the appropriate grid level (see [1] for the actual formulae and relaxation algorithm).

The dummy unknowns (shown in Figure 3(b)) are not updated by the smoothing operation. These values are determined by interpolation when the variables are transferred from the finer or coarser grid (depending upon the stage of the multigrid cycling). These values are set at the beginning of the relaxation phase and act as boundary values while the genuine variables, which are now separated from the interface, are updated. The interpolation formulae used to define these dummy variables are given in the next section. Note, however, that these latter variables are updated in the course of our multigrid cycling so that, at convergence, both our discretized differential equations and our interface conditions are satisfied simultaneously.

6. Interface Conditions

Our standard transfer operators use linear interpolation for all variables. This works reasonably well, although for optimal performance we should use higher-order interpolation the first time we visit a new grid [9]. However, our numerical experiments show that our strategy works well when there is no grid refinement, and it is reasonably simple to implement.

If there is no grid refinement, the transfer operators do not affect the accuracy of the discretization: at convergence we solve the nonlinear, fine-grid equations independent of the transfers. At interfaces, however, the dummy variables do affect the solution because they are, implicitly, boundary values for the two subdomains. It is natural to generate these values by using standard

transfer operators. However, this approach can lead to two types of difficulty.

The primary consideration is that there is no solution to the discrete equations unless discrete mass is conserved. If we consider the flux across the (horizontal) interface, we require

$$2hV(j) = hv(j-1) + hv(j+1),$$

where we have assumed that the fine grid size is h and that the nomenclature is as defined in Figure 4. This requirement is automatically satisfied by our standard restriction operation. However, normal linear prolongation does not have this property. We have modified our prolongation operator so that mass is conserved by this operator as well. We define the fine-grid neighbors of a coarse-grid velocity by

$$v(j+1) = V(j) + (V(j+1) - V(j-1))/8,$$

and

$$v(j-1) = V(j) - (V(j+1) - V(j-1))/8.$$

These expressions are formally second-order accurate, and insure satisfaction of the continuity equation across the interface. It is interesting to note that there are no interpolation formulae based on Lagrange polynomials that satisfy the discrete conservation property and that have order higher than two.

The other aspect of the interface treatment is its effect on accuracy. Taking finite differences of values obtained by interpolation is, in general, rather suspect. This is exactly what we do, since the momentum equation at the interface involves differences of "dummy variables" obtained by interpolation. One can show that with the linear interpolation our treatment of advection terms is first-order accurate at interfaces, while the treatment of viscosity terms is zeroth-order accurate. Global truncation order is ordinarily one order of accuracy better, implying global second-order accuracy for advection-dominated problems, but only first-order accuracy when viscosity terms dominate.

Quadratic interpolation improves accuracy by one power of h, giving consistent treatment of viscosity terms as well at interfaces. Thus, with quadratic interpolation, we can expect second-order global accuracy, even when viscosity terms dominate, assuming there are only a bounded number of interfaces in the domain. Even in the case where the number of interfaces grows without bound, advection-dominated problems remain second-order accurate. All of this suggests that the quadratic interpolation is superior; in practice, one sees relatively little difference. The reason seems to be that for the advection-dominated problems considered, treatment of viscosity terms is relatively unimportant. The simple linear interpolation scheme already gives global second-order accuracy in most cases of interest and was used in most of our experiments, since it is computationally more efficient. However, it is important to note that our uniform grid calculations are second-order accurate. Since our adaptive calculations are substantially more accurate for a given computational cost, the issue of formal truncation error is moot.

7. Data Structures and Related Issues

From the outset of this work we adopted a refinement strategy that would allow any quadrant of any patch to be refined if it satisfied some criterion. When a quadrant is refine, each mass control volume is divided into four, so that our fundamental data objects remain identical, even though they have different mesh spacing and are associated with a different level in the grid hierarchy. The entire patch structure that we generate is described by a quad tree.

Multigrid algorithms for nonlinear PDEs (full approximation storage schemes) calculate a quantity known as the *defect* as an integral part of the iteration strategy. One definition is

$$\tau_h^H = I_h^H \left[L^h u^h \right] - L^H \left[I_h^H u^h \right],$$

which may be thought of as an approximation to the truncation error on the coarse grid. Other investigators have used simpler error estimators based on (say) velocity gradients or curvature; however, we prefer to use the *defect*, since it does not bias particular terms in the differential equation.

Throughout this paper we have refined any quadrant that satisfies

$$\left[\tau_h^H \right]_{quadrant} \geq \left\{ \frac{1}{nVis} \left(\sum_{nVis} \left[\tau_h^H \right]^{power} \right) \right\}^{\frac{1}{power}},$$

where *nVis* is the number of quadrants that are visible (i.e., unrefined). Normally, we take *power* = 1.0. However, this criterion is not crucial to our algorithm, and other criteria can be substituted with minimal effort.

In addition to this criterion, two other mechanisms are used to control refinement. To ensure that our grid ratio does not grow unreasonably, we force a refinement if the ratio is (locally) greater than two. Moreover, in the immediate neighborhood of a re-entrant corner, if we subdivide one quadrant that *touches* the corner, then we refine all three quadrants.

A number of data structures are required to support the various numerical processes that have to be performed during our iterations. The proper design of these data structures enables reasonably simple implementation of each of the numerical operations, even though the arrangement of the data is now more complex. Typical lists of patch descriptors include the following:

- The *NEXT* patch at level N. (Linking the patches on each level this way allows all operations to be done by list traversal, rather than by a recursive tree walk.)

- The *NEIGHBORS* of each patch (simplifying data exchange during relaxation and interpolation operations).

- The identifiers of the children of each patch (*KIDS*) (necessary for prolongation).
- The *AUNTS* (neighbors of the parents) of each patch (required for restriction).

Boundary conditions and geometric descriptors are also maintained for each patch, enabling us, in principle, to handle curved boundaries or stretched meshes for resolution of boundary layers.

With this information, the basic numerical operations can be implemented as loops going along all the patches in a particular grid level. A typical code fragment could be

```
ID = LevPt(N)                          ! First patch on Level N
DO I = 1,LevCnt(N)                     ! All patches on Level N
   Num_Op(Patch(ID),Patch(Kid(ID)))    ! Perform numerical process
   ID = Next(ID)                       ! Next patch at this level
ENDDO
```

where *Num_Op* is one of smoothing, prolongation, restriction, exchange of edge data, or application of boundary conditions. It is clear that the normal multigrid cycling controls determine the invocation of these code fragments.

The natural order is to process patches in the order of their creation (which determines their place in the list). All processes (except smoothing) are independent of the ordering of the points. Experiments have shown that the smoothing rate of our relaxation process is insensitive to the ordering.

The aspect ratio of all cells remains constant throughout the refinement process. Of course, grids built of rectangles can easily be used by appropriate definition of the original grid. However, the smoothing rate degrades if the aspect ratio is very different from unity. We are considering using line relaxation to avoid this effect, but some kind of patch sorting is required before this strategy can be implemented.

8. Numerical Results

We wish to focus attention on three main points. First, the rate of convergence and accuracy that we achieve on our automatically adapted grids must be comparable with our standard multigrid calculations. Second, our local truncation error estimate $\left[\tau_h^H\right]$, which we use to control our local refinement strategies, must give us reasonable patterns. Third, the accuracy we achieve must be comparable with that obtained by other benchmarks. Additionally, we shall discuss briefly our interface treatment and demonstrate the geometric flexibility of the algorithm. However, we have not attempted to develop optimal strategies for all aspects of this work, but rather to validate the basic components.

8.1 The Driven-Cavity Problem

The first example we consider is the driven-cavity problem. This widely known problem is well documented in the literature. We have chosen this problem because it is geometrically simple and the solution is well understood. However, the presence of singularities in the solution clearly is going to cause any adaptive code difficulties. This fact should be borne in mind when considering these results.

The problem domain is a unit square, and the flow is driven by specifying $u = 1$ and $v = 0$ on the top wall. On the other walls, we specify $u = v = 0$. There are singularities in the horizontal velocity in the top corners of the domain.

Table 1(a). Effect of Different Interface Treatments with Re = 1, Fixed Refinement Patterns, and First-Order Boundary Treatment (timings on a Sun/3 with Weitek floating-point accelerator)

Re = 1.0	Linear Interface					Quadratic Interface				
Grid	τ_w	error	ρ^f	#	T (sec)	τ_w	error	ρ^f	#	T (sec)
16	6.09874	2.27(-1)	.41	18	4	6.09874	2.27(-1)	.41	18	4
16+5	5.98808	1.16(-1)	.44	19	11	5.96708	9.52(-2)	.46	19	11
16+5+5	5.96703	9.51(-2)	.46	21	24	5.90704	3.51(-2)	.46	21	24
16+5+5+5	5.94360	7.17(-2)	.45	22	50	5.87535	3.45(-3)	.44	22	51
16+5+5+5+5	5.93034	5.84(-2)	.46	24	107	5.86125	1.06(-2)	.45	24	110
512x512	5.87190	-	-	-	-	5.87190	-	-	-	-

Table 1(b). Effect of Different Interface Treatments with Re = 400, Fixed Refinement Patterns, and First-Order Boundary Treatment

Re = 400	Linear Interface			Quadratic Interface		
Grid	τ_w	Error	#	τ_w	Error	#
64 x 64	2.549(-2)	5.2(-4)	15	2.549(-2)	5.2(-4)	15
64 x 64+5	2.503(-2)	0.6(-4)	12	2.493(-2)	0.4(-4)	12
64 x 64+10	2.503(-2)	0.6(-4)	11	2.500(-2)	0.3(-4)	12
128 x 128	2.479(-2)	1.8(-4)	11	2.479(-2)	1.8(-4)	11
128 x 128+5	2.491(-2)	0.6(-4)	10	2.479(-2)	1.8(-4)	11
128 x 128+10	2.491(-2)	0.6(-4)	9	2.487(-2)	1.0(-4)	10
64 x 64+5+5	2.515(-2)	1.8(-4)	-	2.493(-2)	0.4(-4)	11

Notes: 64 x 64 and 128 x 128 denote uniform-grid calculations.
64 x 64+5 has the five rows of mass control volumes at the top of the domain subdivided (each cell becoming four finer cells).
64 x 64+5+5 has the top five rows of cells of the 64 x 64+5 grid subdivided.
Other definitions are obvious generalizations of this.

First, we consider the effect of the interface treatment on the accuracy of the solution. Tables 1(a) and 1(b) show the wall shear stress and the associated errors at the midpoint of the wall. The wall shear value shown in the tables was calculated by using third-order, one-sided interpolation. The error was measured by taking the value calculated on a 512 x 512 grid as exact. (The discretized equations used a one-sided, first-order approximation for the wall shear, for simplicity.) For the purposes of this comparison only, we have forced the refinement patterns. The grid refinements, explained in the table notes, were chosen to allow reasonable representations of the boundary layer.

From these tests (and others at even higher Reynolds numbers), we deduced that second-order interpolation was adequate for our fully adaptive strategy. (As we expected, the quadratic formulae are somewhat more accurate, but the gains do not seem to justify the additional complexity.) Clearly, this approach differs from that described in, for example, reference [6]. Our goal is to introduce refinements that efficiently reduce the errors associated with the existing grid calculations, rather than to maintain a particular order of truncation error. Formal truncation error analysis and its relationship to actual errors is a complex issue in this context. We are not claiming that our adaptive discretization is second order. However, our uniform grid calculations are, in general, second order. Throughout this section we compare the adaptive and the uniform (i.e., second-order) strategies on the basis of speed, accuracy, and storage.

Table 1(c) shows the performance of the fully automatic adaptive code, relative to uniform grid calculations. The Reynolds number is 400. The errors have been estimated by considering

the wall shear stresses at the midpoint of the moving lid. We have, of course, deliberately chosen a measure that is not in a neighborhood of the corner singularities. Winters and Cliffe [5] have used two independent formulations for their computations (both primitive variable and stream-function/vorticity). The values they give, $\tau_w|_{x=.5}$ = 2.661(-2) or 2.481(-2), can be used as an estimate of the accuracy of their figures. It is clear that both our uniform and adapted grids converge to similar answers. However, our present refinement strategy concentrates on the singularities in the corners. The wall shear in the middle of the lid does not appear to be especially sensitive to the corner treatment. (This seems reasonable: it is in the middle of a boundary layer driven by the moving wall.) Therefore, the adaptive calculations will be somewhat less effective than uniform ones for this particular measure of accuracy (see Table 1(c)).

Table 1(c). Driven-Cavity Problem: Wall Shear Stress at Midpoint of Moving Lid with Re = 400

Minimum		Uniform Calc. (2nd order)			Adaptive Calc. (2nd order)		
Grid Size	Level	$\tau_w\|_{x=.5}$	Time (sec)	# Patch	$\tau_w\|_{x=.5}$	Time (sec)	# Patch
1/16	3	0.03601	6	21	0.03601	6	21
1/32	4	0.02825	17	85	0.02937	17	43
1/64	5	0.02517	46	341	0.02793	34	101
1/128	6	0.02505	109	1365	0.02633	66	220
1/256	7	0.02508	828	5461	0.02554	126	462
1/512	8	--	--	--	0.02539	277	954
1/256[a]	7	--	--	--	0.02554	85	462
1/512[a]	8	--	--	--	0.02539	213	954
1/256[b]	7	--	--	--	0.02541	99	540
1/512[b]	8	--	--	--	0.02522	256	1229
1/256[c]	7	--	--	--	0.02516	293	1637

Notes:

F(2,2) cycles, tol=1.0(-7),ref-power=0.35.
Timings on an IBM Risc System/6000® (model 530).

[a] results obtained with the *ad hoc* cycling strategy described at the end of Section 8.2.
[b] results obtained with ref-power = 0.2 and the *ad hoc* cycling strategy (ref-power = 0.35 otherwise).
[c] results obtained with the *ad hoc* cycling strategy described below on an Adapt(5,7) grid.

The vertical velocity extrema at the horizontal mid-plane are an alternative error measure. In Table 1(d) we compare adaptive and uniform calculations on the basis of these quantities. Again, we see the importance of the refinement strategy. However, the Adapt(5,7) calculation has accuracy comparable with the uniform seven-level calculation (at least for the minimum vertical velocity) with considerable savings in cpu time and computer storage. Physically, one could argue that the downturn in the horizontal boundary layer is caused by the pressure maximum at the top right-hand corner of the domain. Our adaptive strategy has captured this downturn and has also refined the mesh downwards, following this "jet." In Figure 5, we show a seven-level adaptive grid and the resulting flow visualization consisting of the level curves for the stream function.

Table 1(d). Driven-Cavity Problem: Velocity Extrema at the Horizontal Midplane

Minimum Grid Size	Level	Comment	Midplane Y=0.5 V_{min}	Midplane Y=0.5 V_{max}	Time (sec)	# Patch
1/16	3	Uniform	-0.2931	0.1801	6	21
1/32	4	Uniform	-0.3923	0.2529	17	85
1/64	5	Uniform	-0.4396	0.2946	46	341
1/128	6	Uniform	-0.4518	0.3021	109	1365
1/256	7	Uniform	-0.4534	0.3033	828	5461
1/256[a]	7	Adapt(3,7)	-0.4388	0.2852	85	462
1/256[a]	7	Adapt(5,7)	-0.4523	0.3020	293	1637

Notes:

Validation and serial performance for the adaptive code: vertical velocity extrema across the horizontal midplane.
Adapt(n,m) defines a calculation with n levels of uniform refinement and then adaptive refinement up to level m. (In this table we had ref-power = 0.35.)
Timings on an IBM Risc System/6000® (model 530).

[a] results obtained with the *ad hoc* cycling strategy described at the end of Section 8.2.

Table 1(e) shows the rates of convergence for the driven-cavity problem on uniform and adapted grids. As is typical of multigrid algorithms on nonlinear problems, the convergence rate varies somewhat as the number of levels increases but remains bounded well away from one uniformly in the number of grid levels.

Of greater significance, we note that the convergence rates achieved on the adapted meshes are comparable to the uniform grid rates. One would not necessarily expect this result on theoretical grounds; it amounts to a pleasant surprise. The calculations here are terminated on each grid when the norm of the residual is less than 10.0(-07). The reduction factors given in the table are the *final reduction factors*; in some cases they have not reached their asymptotic values.

Each of our test cases contains some form of singularity (or singularities). We are not treating these in any special fashion; they are automatically handled by our refinement strategy. In general, for complex flows, it is impossible to predict all possible singularities; therefore, methods that rely on accurate and special treatment of these regions are not sufficiently flexible. Our solutions are obviously going to be poor in small neighborhoods of the singularities, but this effect is local; moreover, careful mesh grading allows a solution to be obtained without degradation of the order of accuracy (see, for example, [3] and [8]).

Table 1(e). Driven-Cavity Problem:
Rate of Convergence on Adaptive and Nonadaptive Grids, with Re = 400

	Uniform Calculation (2nd)				Adapted Calculation (2nd)			
Grid Level	Total Pats.	Fine Pats.	No. Cycles	ρ^f	Total Pats.	Fine Pats.	No. Cycles	ρ^f
2	5	4	23	.51	5	4	23	.51
3	21	16	20	.47	21	16	20	.47
4	85	64	24	.52	43	22	19	.42
5	341	256	22	.55	101	52	17	.23
6	1365	1024	15	.31	220	102	16	.47
7	5461	4096	20	.62	462	169	15	.42
8	--	--	--	---	954	321	17	.52

Notes:

F(2,2) cycles, tol. = 1.0(-7), ref-power = 0.35.
Second-order boundary conditions.

8.2 The Flow over a Backward-Facing Step Problem

The flow over a backward-facing step is another geometrically simple and well-documented problem. As is usual, we have specified a parabolic inlet profile (fully developed flow), and we have fully developed outlet conditions. No slip conditions are imposed at all walls. In this configuration, we calculate the flow upstream of the step, thereby solving for the flow around the step and not ignoring the singularity at the corner. In contrast to [15], important features of this flow are the recirculation zone behind the step and the velocity profiles downstream of the step.

In Table 2(a) we give the convergence rates for the uniform-grid and adapted-grid calculations. Little degradation is observed in the adapted-grid computations, despite the complexity of the problem domain. However, the uniform-grid calculations at level 3 converge more quickly

because the diffusion terms become more important than the convective terms as the mesh-size decreases. This effect is not seen on the adaptive calculations because some parts of the coarser grid remain visible.

Table 2(a). Rate of Convergence for the Backward-Facing Step Problem: Uniform and Adaptive Calculations

	Uniform Calculation				Adapted Calculation			
Grid Level	Total Pats.	Fine Pats.	No. Cycles	ρ^f	Total Pats.	Fine Pats.	No. Cycles	ρ^f
2	180	144	16	.52	180	144	16	.52
3	756	576	9	.24	240	60	11	.41
4	--	--	--	--	391	114	10	.40
.								
9	--	--	--	--	525	19	14	.42
10	--	--	--	--	544	19	15	.42

Notes: F(2,2) cycles with Re = 150, power = 2.0, and tol. = 10.0(-6).

Table 2(b). Results for the Backward-Facing Step:
Uniform and Adaptive Calculations and Benchmark

	Velocity				
--	Cliffe	(2,2)	(3,3)	(2,4)	(2,10)
U_{max}					
1.6 step-ht	0.904	0.897	0.903	0.906	0.905
4.0 step-ht	0.725	0.725	0.722	0.726	0.725
U_{min}					
1.6 step-ht	-0.102	-0.103	-0.103	-0.102	-0.101
4.0 step-ht	-0.049	-0.036	-0.047	-0.039	-0.038
V_{max}					
1.6 step-ht	0.010	0.010	0.010	0.010	0.010
4.0 step-ht	0.0	0.0	0.0	0.0	0.0
V_{min}					
1.6 step-ht	-0.070	-0.068	-0.070	-0.071	-0.071
4.0 step-ht	-0.091	-0.077	-0.087	-0.089	-0.089
No. patches	--	180	756	391	544
Time (sec)	--	73	77	113	--
Total Time (sec)	--	73	77	278	--

Notes: (2,2): uniform 2-level calculation with mesh size = step-height/8.
(3,3): uniform 3-level calculation with mesh size = step-height/16.
(2,N): adapted calculation with maximum mesh size = step-height/8 and minimum mesh size = step-height/$2^{(N+1)}$.

To assess the accuracy of our approach, we have included uniform, adapted, and benchmark solutions; see Table 2(b). The benchmark solutions were taken from reference [14]. Clearly, all the sets of calculations are in agreement. Moreover, the adaptive calculations with two levels of

automatic refinement (shown as (2,4) in the table) use only half the storage of the uniform 3-level calculations for comparable accuracy. A weak singularity in the neighborhood of the corner inhibits further improvement of the solution with our current refinement strategy. However, there is no *degradation* in the solution if we allow more refinement (we allowed a further six levels). Therefore, our interface treatment is reasonable.

We show two sets of flow visualizations and refinement patterns for this configuration. The first set (Figure 6(a)) was obtained by using a naive treatment of the inlet and first-order treatments of the walls. The code has *automatically* detected this and refined the pattern in the areas where the approximation is poor. When we improved these two aspects so that we had a consistently second-order treatment throughout the domain, we obtained the refinement patterns shown in Figure 6(b) (the values used in Table 2(b) used the improved treatments).

Our coarsest-grid treatment simply consists of many sweeps. In many circumstances this is adequate. However, in cases where there are many patches on the first grid, it is obvious that various forms of acceleration could be used. At present, these have not been implemented since the benefits are probably marginal. Moreover, we can decrease this effect by making minor modifications to the multigrid cycling strategy.

Another factor that affects our adaptive timings is that, at present, we solve a *sequence* of flow problems so that we have reliable error estimates that are "uncontaminated" by insufficient numerical convergence; these are the basis of our refinement strategy. It may be possible to avoid this sequential procedure by doing the refinement and solution concurrently (in some sense). However, we have avoided this strategy in the present study since the convergence errors confuse the refinement patterns. The only attempt we have made to address this issue is the *ad hoc* strategy referred to in the tables. Here we do the following: when performing an *Adapt*(n,m) calculation (uniform refinement on n grids, then adaptive refinement up to level m), we reset the cycling tolerance to its square root until we have completed the preliminary problems and we have an adapted m-level grid. We then solve the m-level problem to the full tolerance. This strategy decreases the set-up time, without significantly changing the final adapted grid or the computed solution. For two of the geometries in this paper, this strategy is quite effective. However, for the backward-facing step geometry we were able to reduce the time for the *Adapt*(2,4) calculation only to 180 seconds. The coarsest-grid effects still dominate when we have such a small number of levels.

8.3 The Sudden Expansion/Contraction Geometry

The third example is new. It consists of a sudden expansion/contraction and was chosen to show the flexibility of our approach. (The choice was also tempered by the desire to have an intuitive feeling for the correctness of our results.) The geometry is shown in Figure 7. At the

extreme left, we specify a parabolic inlet profile for the horizontal velocity (and $v = 0$). On the right we specify fully developed flow conditions: $v = 0$ and $\frac{\partial u}{\partial x} = 0$. Everywhere else we have $u = v = 0$. This flow has *four* re-entrant corners. An interesting quantity is the flux across the "entrance" to the expansions in the domain.

The rates of convergence (Table 3) agree with the results of the preceding sections: adaptivity does not slow down convergence. Again we see the uniform-grid calculations improving as the mesh becomes finer, because the elliptic terms become dominant; but this effect is not seen in the adaptive calculations, because sections of the coarse grids remain "visible" and limit the rate of convergence. The flow visualization (Figure 7) shows that we are obtaining qualitatively reasonable results. There are two shear-driven recirculation cells formed behind the steps. The streamlines seem to be quite reasonable.

The streamlines shown in the figures were calculated by integrating over the adapted grid. The use of this adapted mesh was necessary because the conservation equation is satisfied here. We used Gaussian quadrature, modified to capture the compound grid automatically. Techniques that ignored this fact gave disappointing results.

One quantity of interest is the transport from the main channel into the two expansions. We have measured this by looking at the vertical velocity extrema along the "continuation" line across the top of the channel. Both the uniform and adaptive calculations converge to the same values; however, when the *ad hoc* cycling strategy is used, the adaptive calculations are faster and use less memory.

Table 3. Sudden Expansion/Contraction Geometry:
Rate of Convergence and Vertical Velocity Extrema

	Uniform Calculation					Adaptive Calculation					
Minimum Grid Size	Vert. Velocity Max	Vert. Velocity Min	Time (sec)	No. Patches	No. Cycles	Vert. Velocity Max	Vert. Velocity Min	Time (sec) Item	Time (sec) Cumul.	No. Patches	No. Cycles
1/8	0.0212	-0.073	6	25	11	0.0212	-0.073	6	6	25	11
1/16	0.0224	-.088	9	105	9	0.0230	-0.090	11	16	55	14
1/32	0.0233	-.092	22	425	8	0.0243	-0.094	18	34	111	12
1/64	0.0254	-.099	72	1705	8	0.0254	-0.100	27	61	185	14
1/128	---	---	--	--	--	0.0262	-0.100	44	106	283	14
1/64[a]	---	---	--	--	--	0.0257	-0.100	31	46	186	14
1/128[a]	---	---	--	--	--	0.0262	-0.100	44	69	284	14

Notes:

Re = 50, F(2,2) cycles, tol=1.0(-7), ref-power=1.0.
Timings on an IBM Risc System/6000® (model 530).

[a] Results obtained with the *ad hoc* cycling strategy.

9. Discussion

There are three reasons why one is interested in automatic adaptive gridding strategies: speed, storage, and reliability of the solution. The first item has been dealt with earlier.

The issue of storage is fairly simple, and we have been able to achieve significant savings in all cases of interest. Here we simply add the comment that, in three-dimensional problems, the question of efficient storage utilization is significantly more important and the benefits from adaptive gridding correspondingly greater.

The issue of reliability of the computed solution is complex. Our present refinement strategy is somewhat *ad hoc*, and a detailed study of refinement strategies is beyond the scope of this paper. However, our examples illustrate three types of error: (1) those induced by some type of singularity, (2) those caused by comparatively poor numerical treatments (e.g., low-order boundary formulae), and (3) those caused by insufficient grid resolution. The best that one can hope to do in the presence of singularities is to treat them sufficiently well that their effect is local

(unless one uses some semianalytic approach). Our code does this. Moreover, the other two effects are detected and automatically reduced by our refinement strategy until the errors are dominated by singularity effects.

The question of how a numerical solution varies with respect to "small" changes in the grid is not well understood, nor is the global effect of singularities in the domain. Attention must be given to these points before we can achieve "full reliability." However, it is interesting to note that the problems with singularities are not restricted to error estimators based on the defect (or the local truncation error). Richardson extrapolation techniques and simpler gradient estimators can have the same problems.

Throughout this work we have measured errors by estimating a *continuous* solution and defining errors with respect to this. We have the following equation:

$$L^h \left[e^h \right] \approx \tau_h^H,$$

where e^h is the discrete error. Since the equation is linear, it can be solved with comparatively little extra work, given that the operator and the gridding are already defined. This approach offers possibilities of improving our refinement strategies. However, e^h may contain nonlocal errors, thereby complicating its use in a refinement strategy.

10. Conclusions

As is customary with this type of work, we have demonstrated the effectiveness and accuracy of our techniques by considering a few, well-known model problems.

We have shown that the rate of convergence we observe for the adaptive calculations does not degrade significantly compared to the uniform ones. Our estimate of the local truncation error (τ_h^H) gives us qualitatively correct refinement patterns, which are capable of detecting deficiencies in the numerical treatment as well as in the grid. The accuracy of our results is comparable with that of other benchmarks for both the uniform and the adaptive calculations. Our current adaptive strategy will deal with all three classes of error. However, it continues to "home in" on singularities, even though this is not always an optimal strategy.

Additionally, we have shown that the interpolation that we use at interfaces is good and that we are easily able to handle a range of geometries. For some of the cases we have considered, we have been able to achieve significant gains in speed and storage compared to uniform grid calculations. However, we have shown that this is not always the case.

11. Future Work

In the current work we showed that we can separate the three components--the multigrid algorithm, the adaptive strategy, and the numerical approximations--as was our design goal. Now that we have done this, we can modify and improve each of these aspects separately. In particular, we are looking at efficient ways to improve the generality, scope, and accuracy of the algorithm. We are also focusing our attention on parallelization issues. A shared-memory version of this code is running, and we are about to look at implementation on scalable distributed-memory architectures, such as the Intel® Touchstone DELTA system.

Acknowledgments

This work benefited from several fruitful discussions with Dr. N. W. Wilkes of Harwell Laboratory, England, who provided important assistance with our boundary treatments. We are happy to acknowledge the assistance of Lisbeth Skodvin, Bergen Scientific Centre, for help with the visualizations used in this paper.

IBM and Risc System/6000 are trademarks of IBM Corporation. Intel is a trademark of Intel Corporation.

References

[1] C. P. Thompson, G. K. Leaf, and S. P. Vanka, Application of a multigrid method to a buoyancy-induced flow problem, in *Multigrid Methods Theory, Applications, and Supercomputing*, S. F. McCormick (Ed.), pp. 605-630, Marcel Dekker, New York, 1988.

[2] H. C. Ku, R. S. Hirsh, and T. D. Taylor, A pseudospectral method for solution of the three-dimensional incompressible Navier-Stokes equations, *J. Comp. Phys.* Vol. 70, pp. 439-462, 1987.

[3] G. Strang and G. J. Fix, *An analysis of the finite element method*, Prentice-Hall, Englewood Cliffs, 1973.

[4] A. Brandt and N. Dinar, Multigrid solutions to elliptic flow problems, in *Numerical Methods for PDEs*, S. V. Parter (Ed.), pp. 53-147, Academic Press, New York, 1979.

[5] K. H. Winters and K. A. Cliffe, A finite element study driven laminar flow in a square cavity, AERE - R 9444, AERE Harwell, UK, 1979.

[6] G. Chesshire and W. D. Henshaw, Composite overlapping meshes for the solution of partial differential equations, *J. Comp. Phys.* Vol. 90, No. 1, pp.1-64,1990.

[7] S. P. Vanka, Block-implicit multigrid solution of Navier-Stokes equations in primitive variables, *J. Comp. Phys.* Vol. 65, No. 1, pp.138-158,1986.

[8] L. Fuchs, A local mesh refinement technique for incompressible flows, *Computers and Fluids*, Vol. 14, No. 1, pp. 69-81, 1986.

[9] A. Brandt, Multigrid Techniques: 1984 Guide with Applications to Fluid Dynamics, *Computational Fluid Dynamics, Lecture Series 1984-04*, von Karman Institute, Belgium, 1984.

[10] K. Stueben and U. Trottenberg, Multigrid Methods, Fundamental Algorithms, Model Problem Analysis and Applications, *Multigrid Methods, Lecture Notes in Mathematics*, Vol. 960, Springer Verlag, Berlin, 1982.

[11] D. Bai and A. Brandt. Local mesh refinement multilevel techniques *SIAM J. Sci. Stat. Comput.* Vol. 8, pp. 109-134, 1987.

[12] S. McCormick and J. W. Thomas. The fast adaptive composite grid (FAC) method for elliptic equations, *Math. Comp.* Vol. 46, pp. 439-459, 1986.

[13] N. S. Wilkes and C. P. Thompson. An evaluation of higher-order upwind differencing for elliptic flow problems, in *Numerical Methods for Laminar and Turbulent Flow*, C. Taylor, J. A. Johhnson and W. R. Smith (Eds.), pp. 248-257, Pineridge Press, 1983.

[14] K. A. Cliffe, I. P. Jones, J. Porter, C. P. Thompson and N. S. Wilkes. Laminar flow over a backward facing step: Solution to a test problem, in *Analysis of Laminar Flow over a Backward-Facing Step: A GAMM Workshop*, K. Morgan, J. Periaux, F. Thomasset (Eds.), pp. 140-161, Friedr. Vieweg and Sohn, 1983.

[15] M. C. Thompson and J. H. Ferziger, An adaptive multigrid technique for the incompressible Navier-Stokes equations, *J. Comp. Phys.* Vol. 82, No. 1, pp. 94-121, 1989.

[16] C. P. Thompson, W. R. Cowell, and G. K. Leaf. On the parallelization of an adaptive multigrid algorithm for a class of flow problems, preprint MCS-P205-0191, Mathematics and Computer Science Division, Argonne National Laboratory, 1991.

[17] L. P. Hackman, G. D. Raithby, and A. B. Strong, Numerical predictions of flows over backward-facing steps, *Int. J. Num. Meth. Fluids*, Vol. 4, pp. 711-724, 1984.

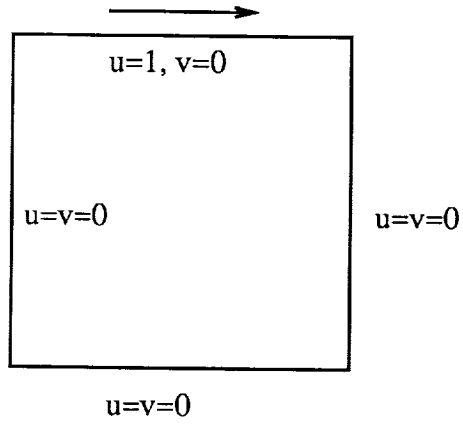


Figure 1(a). Driven Cavity: Solution Domain and Boundary Conditions

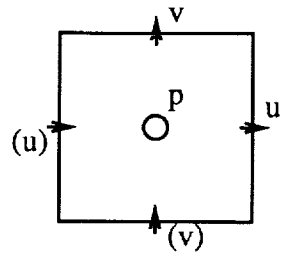
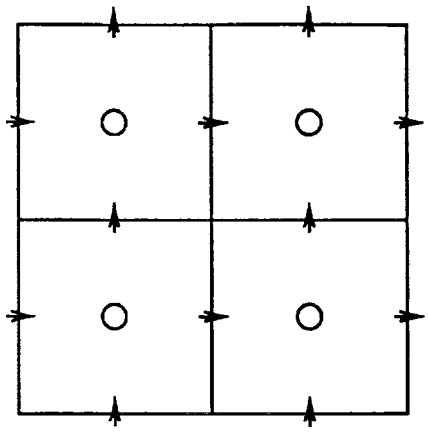
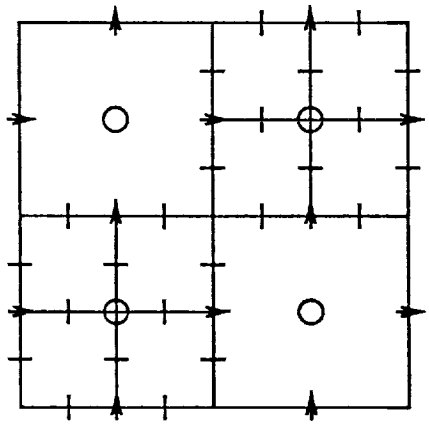


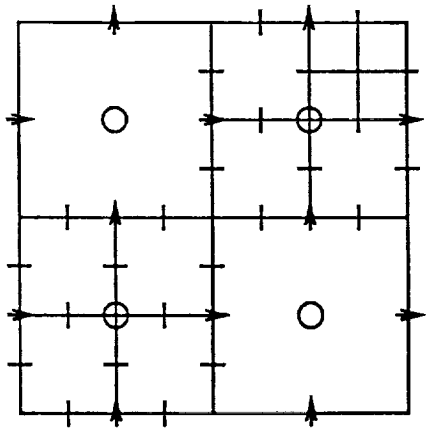
Figure 1(b). Staggered Grid: Location of Variables. Bracketed variables are associated with adjacent cells.



(a) One grid level, one patch.



(b) Two grid levels, three patches.



(c) Three grid levels, four patches.

Figure 2. Composite Grids at Each Level

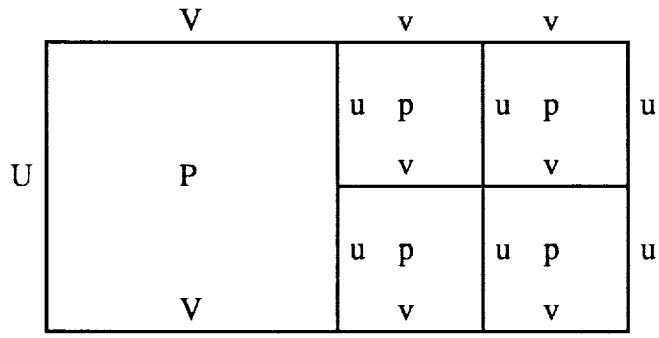


Figure 3(a). The Canonic Problem: One Coarse-Grid Patch Interfacing with Four Fine-Grid Patches. This tiling is sufficient to generate all configurations of interest. We show the location of "genuine" unknowns on the compound grid. At these locations we solve approximations of the differential equations. (Coarse-grid variables are in upper case, fine-grid ones in lower case.)

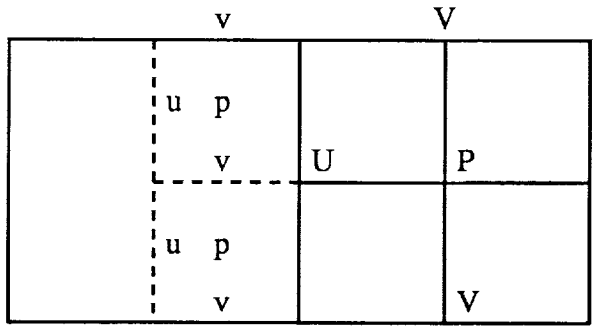


Figure 3(b). The Canonic Problem Modified to Show the Dummy Unknowns Introduced for Computational Convenience. The variables shown here are not unknowns in our system of equations but dummy variables that are introduced for convenience. Special, additional equations are introduced for them. (Coarse-grid variables are in upper case, fine-grid ones in lower case.)

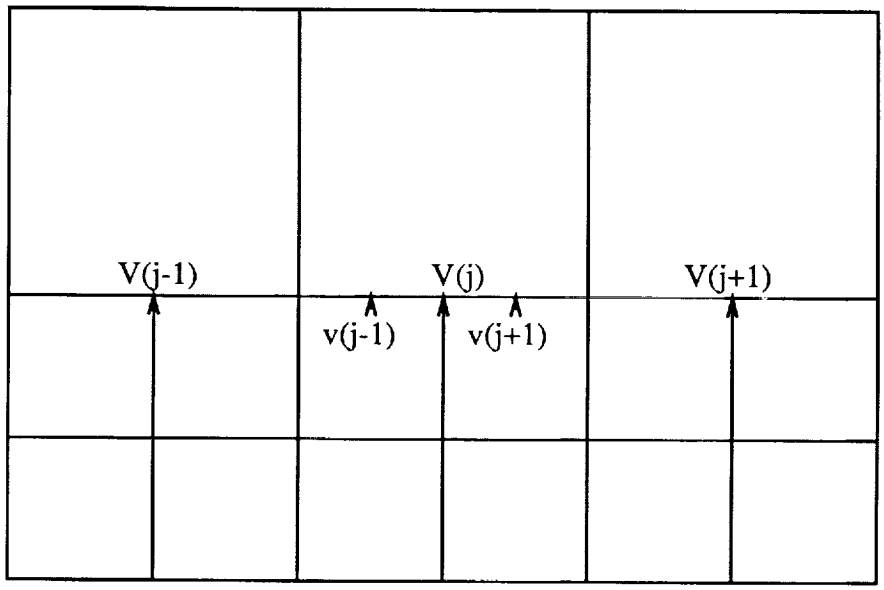


Figure 4. Prolongation at Interfaces. (Coarse-grid unknowns are shown in upper case, fine-grid ones in lower case.)

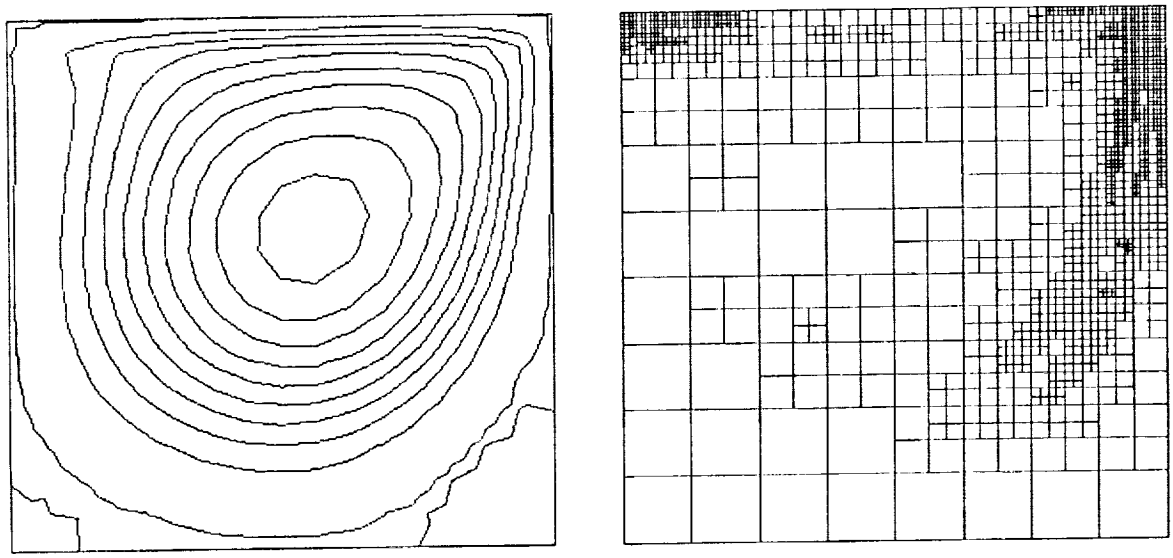


Figure 5. The Driven Cavity Problem: Flow visualization, an adapted grid and problem definition. On the top wall we drive the flow by specifying $u = 1$ (and $v = 0$). On the other walls we have $u = v = 0$. There are singularities in the horizontal velocity in the two top corners of the domain.

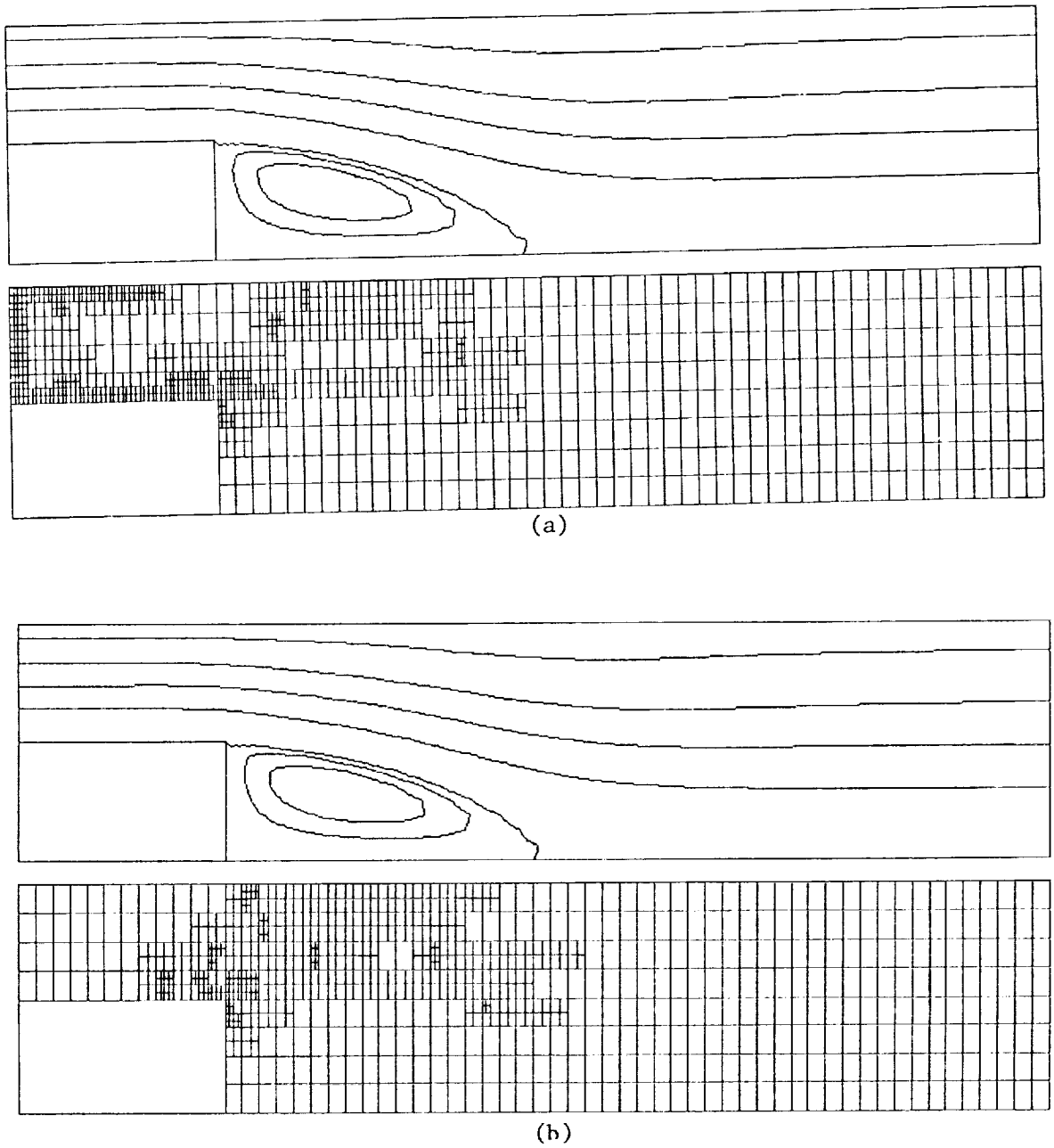


Figure 6. The Backward-Facing Step Problem: Flow visualization, an adapted grid and problem definition.
(a) First-order boundary conditions and standard inlet treatment,
(b) Second-order boundary conditions and modified inlet treatment.
At the extreme left we specify a parabolic inlet profile for the horizontal velocity (and $v = 0$). On the right we specify fully developed flow conditions: $v = 0$ and $\frac{\partial u}{\partial x} = 0$. On all other boundaries we have $u = v = 0$. Notice that we are solving the flow *around* the step and not ignoring the singularity at the corner. This is a significantly more difficult problem than considering a rectangular domain only.

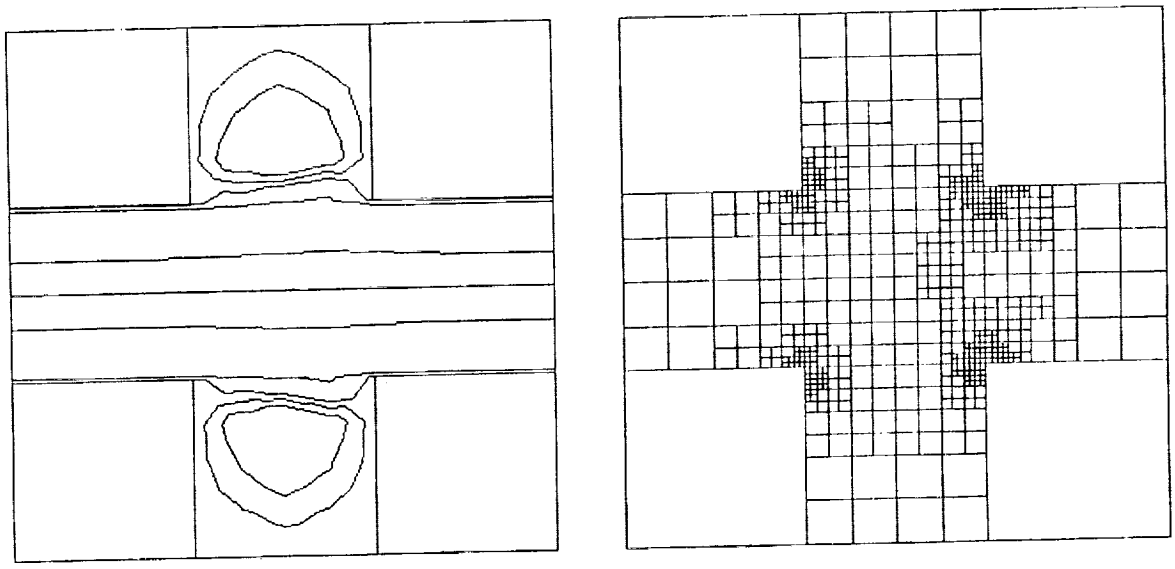


Figure 7. The Cross Geometry: Flow visualization, an adapted grid, and problem definition. At the extreme left we specify a parabolic inlet profile for the horizontal velocity (and $v = 0$). On the right we specify fully developed flow conditions: $v = 0$ and $\frac{\partial u}{\partial x} = 0$. On all other boundaries we have $u = v = 0$. This flow has *four* re-entrant corners. An interesting quantity is the flux across the "entrance" to the expansions in the domain. Peak velocities are shown in Table 3.

NASA
National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA CR-187595 ICASE Report No. 91-36		

4. Title and Subtitle

A DYNAMICALLY ADAPTIVE MULTIGRID ALGORITHM FOR THE INCOMPRESSIBLE NAVIER-STOKES EQUATIONS -- VALIDATION AND MODEL PROBLEMS

5. Report Date

June 1991

6. Performing Organization Code

7. Author(s)

C. P. Thompson
G. K. Leaf
J. Van Rosendale

8. Performing Organization Report No.

91-36

10. Work Unit No.

505-90-52-01

9. Performing Organization Name and Address

Institute for Computer Applications in Science and Engineering
Mail Stop 132C, NASA Langley Research Center
Hampton, VA 23665-5225

11. Contract or Grant No.

NAS1-18605

13. Type of Report and Period Covered

Contractor Report

12. Sponsoring Agency Name and Address

National Aeronautics and Space Administration
Langley Research Center
Hampton, VA 23665-5225

14. Sponsoring Agency Code

15. Supplementary Notes

Langley Technical Monitor:
Michael F. Card

Submitted to J. Applied Numerical Mathematics

Final Report

16. Abstract

We describe an algorithm for the solution of the laminar, incompressible Navier-Stokes equations. The basic algorithm is a multigrid method based on a robust, box-based smoothing step. Its most important feature is the incorporation of automatic, dynamic mesh refinement. Using an approximation to the local truncation error to control the refinement, we use a form of domain decomposition to introduce patches of finer grid wherever they are needed to ensure an accurate solution. This refinement strategy is completely local: regions that satisfy our tolerance are unmodified, except when they must be refined to maintain reasonable mesh ratios. This locality has the important consequence that boundary layers and other regions of sharp transition do not "steal" mesh points from surrounding regions of smooth flow, in contrast to moving mesh strategies where such "stealing" is inevitable.

Our algorithm supports generalized simple domains, that is, any domain defined by horizontal and vertical lines. This generality is a natural consequence of our domain decomposition approach. We base our program on a standard staggered-grid formulation of the Navier-Stokes equations for robustness and efficiency. To ensure discrete mass conservation, we have introduced special grid transfer operators at grid interfaces in the multigrid algorithm. While these operators complicate the algorithm somewhat, our approach results in exact mass conservation and rapid convergence.

In this paper, we present results for three model problems: the driven-cavity, a backward-facing step, and a sudden expansion/contraction. Our approach obtains convergence rates that are independent of grid size and that compare favorably with other multigrid algorithms. We also compare the accuracy of our results with other benchmark results.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
adaptive grids, incompressible flow	64 - Numerical Analysis Unclassified - Unlimited

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	32	A03