JOURNAL OF COMPUTATIONAL PHYSICS **68**, 283–306 (1987)

An Adaptive-Mesh Finite-Difference Solution Method for the Navier–Stokes Equations*

PAOLO LUCHINI

Istituto di Gasdinamica, Facoltà di Ingegneria, 80125 Napoli, Italy

Received March 19, 1985; revised March 21, 1986

A variable-spacing grid system for finite-difference calculations is presented. The system allows single points to be added to or deleted from the mesh independently of each other, while maintaining each point at the center of a symmetrical cross formed with four other mesh points. The single finite-difference form of the steady, incompressible Navier–Stokes equations necessary for use with this system is written in a suitable form insuring stability, and the addition–deletion procedure is easily automated and transformed into a self-adjusting algorithm capable of recognizing the high-gradient regions of the solution field and selectively refining the mesh in those regions. It also determines the best-suited number of points for the calculation. The method is finally applied to two test cases to show its performance. © 1987 Academic Press, Inc.

1. INTRODUCTION

Much of the recent research effort in computational fluid dynamics has been devoted to the automatic generation of non-uniform grids for finite-difference calculations [1, 2]. Non-uniform grid techniques are dictated in two different situations: when fluid flow fields must be simulated in domains with curved solid boundaries and in cases, typical of high-Reynolds number fluid dynamics, when small regions with very high gradients occur. To be useful, a non-uniform-mesh finite-difference method of solution must be simpler than competing finite-element methods, and the grid must be automatically generated, to avoid the burden of manually collocating thousands of mesh points. To handle most efficiently the high-gradient regions, the grid should also be dynamically self-adjusting, with an automatic mesh refinement wherever these regions develop.

A major trend in research on non-uniform grids has been to use as an intermediate step a coordinate transformation from the physical domain to a calculation domain, where a rectangular grid is drawn, or, equivalently, to introduce in the physical plane a curvilinear coordinate system and place mesh nodes at constant increments of the curvilinear coordinates [2, 3]. The differential equations to be solved are first rewritten in the transformed plane and then approximated by finite-difference expressions. Generally, these methods solve the problem of the automation of grid generation by obtaining the coordinate transformation as the

* Work supported by Ministero Italiano della Pubblica Istruzione.

numerical solution to some elliptic partial-differential equation. While this scheme and its many variants can easily generate a curvilinear coordinate system fitting any boundary shape, its use for selective mesh refinement in high-gradient regions is less straightforward, as it requires the introduction in the mesh-generating elliptic equation of arbitrary functions, which are to be determined more or less by trial and error [2, 3, 5].

Other authors have been using methods in which a substantially one-dimensional coordinate stretching is applied along each direction of a predefined (generally rectangular or polar) coordinate system. These methods (e.g., [6–8], see also [2]) have reached the highest degree of sophistication in grid adaption criteria, but of course much less flexibility is allowed in the mesh movements.

A general limitation of all variable-spacing schemes based upon coordinate transformations is the topological constraint placed upon the distribution of mesh points by the very existence of coordinate lines. This implies that a thickening of the mesh in a region must be accompanied by a depletion in the adjacent regions, so it is not possible to modify a small part of the grid during the calculation without moving a major portion of all the points.

To avoid these disadvantages, we decided to discard grid-generating algorithms based upon curvilinear coordinate systems. We devised a finite-difference scheme where mesh points can be created or deleted in any part of the solution domain, according to proper rules, independently of each other. Therefore the mesh can be easily modified during the iterative calculation and adjusted most closely to the features of the developing solution.

At the basis of our scheme is the well-known observation that a very simple and accurate finite-difference approximation of the incompressible Navier–Stokes equations (and, indeed, of many other equations of mathematical physics) can be written on five points arranged in the shape of a symmetrical cross, whereas if the symmetry is broken much accuracy is lost. With this idea in mind, we sought and found a scheme of creation and deletion rules for grid points having the property that each point is always the center of a symmetrical cross, of varying size, formed with four other points belonging to the grid (with the possible exception of points lying on the boundary of the domain). The same discretized equations can be used for all the points of such a grid, the only variable parameter being cross size, with a gain in speed, programming ease and accuracy.

The grid system is presented in Section 2. Suitable iterative calculation schemes for the Navier-Stokes equations and their boundary conditions are presented in Sections 3 and 4. Automatic mesh-refinement criteria are discussed in Section 5. Calculation examples and tests are contained in Section 6 and 7.

2. Dynamically Adjustable Mesh

The present method applies to those elliptic partial-differential equations like, e.g., the incompressible Navier–Stokes equations, which can be written in such a form where the second derivatives appear only aggregated in the Laplacian operator. A widespread finite-difference approximation of the Laplacian operator is obtained using the values of the unknown function at five points arranged in the shape of a symmetrical cross [9]. This formula has a higher order of approximation than other formulas employing five points. On the same pattern of points first derivatives can be approximated by central differences with second-order accuracy, without restrictions on the form in which they appear in the equations.

To exploit the above properties of the symmetrical cross we sought a variable-spacing mesh in which each point is the center of a symmetrical cross formed with four other points, except at most for points falling on the boundary of the solution domain. We also wanted the possibility to increase or decrease dynamically the number of points in the mesh, by adding or deleting a single point at a time.

We found that such a mesh can be built by the following iterative procedure, provided the presence of crosses with two different orientations is permitted. Assume that at some stage in the process there exists a square pattern in a portion of the mesh where each point is the center of a cross formed with its four closest neighbours. In the center of one cell of this square grid a new point can be added (Fig. 1a), and a cross can be associated with it whose orientation is tilted by 45° with respect to the existing ones and whose vertices are the vertices of the original square. The size of the new cross is $\sqrt{2}$ times smaller than that of the old ones.

After this operation has been repeated four times on the four squares surrounding a point of the initial grid (Fig. 1b), the cross associated with this point in turn can be substituted by a smaller, 45°-tilted, cross having its vertices in the newly generated points. At this stage a square grid pattern is restored (Fig. 1c), even if with a tilted orientation, and the whole process can be repeated indefinitely, giving as fine a mesh as wanted at any particular place. To initiate the process it is sufficient to have a mesh composed of four points forming a square, after which a fifth point can be added in the middle, and then four along the sides, and so on.

By this procedure a variable-spacing mesh, fine where required and coarse

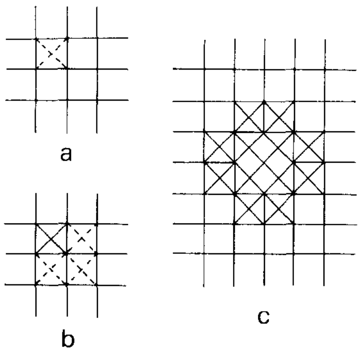

FIG. 1. Steps of the cross addition process.

elsewhere, can be built step-by-step, either interactively, under control of the operator, or automatically, by adjusting the mesh to the behaviour of the unknown function continuously during the calculation.

Of course, points can be as easily removed from the mesh by reversing the steps of the addition process.

In order to avoid the formation of very irregular clusters of points in the mesh, it is useful to add the further constraint that a new point should not be created if the cross associated with its neighbours (i.e., the points that are going to be vertices of the new cross) is larger than $\sqrt{2}$ times the size of the new cross. When this situation occurs, the addition of the new point should be delayed until the size of the neighbours is reduced, or, alternatively, the size of the offending neighbours should be immediately reduced by adding more points as necessary, and then the new point should be created. Conversely, a cross should not be enlarged, for the purpose of deleting points, if it is already larger than the crosses associated with its vertices.

This constraint insures a smooth transition between coarse and fine parts of the mesh and is easily incorporated into the mesh-manipulation procedures; besides providing a smooth matching between the calculations performed on different-size crosses, adherence to this rule simplifies the implementation of the procedures necessary to check whether a point exists already in the mesh before creating a new point at the same position.

In order to maximize the speed of calculation and the ease of adding and deleting points, the mesh can be represented in a computer program as a linked data structure. In particular, each point will be represented by a record containing four links to the four ends of the cross centered in that point, a numeric field containing the distance of the ends from the center, a flag indicating whether the cross is straight-up or tilted, and the numeric values of all the unknowns entering the system of equations to be solved.

To manage the described mesh structure, it is sufficient to prepare two computer procedures, one which, when activated, reduces by $\sqrt{2}$ times the size of any particular cross, adding points around it as needed, and one which enlarges by $\sqrt{2}$ times the size of a cross, deleting the points which become no longer necessary.

The coding of these procedures can profit by the many symmetries of the cross scheme and can be contained, e.g., in about 200 Pascal statements.

A great simplification afforded by the use of crosses with two different orientations is that the difference equations representing the problem must be written for a single kind of point cluster, that is, a symmetrical cross. Since the formulation of a physical problem is generally invariant to rotation, except for the terms representing external interactions, only in the latter terms is it necessary to keep track of the orientation of crosses.

Other finite-difference, variable-spacing methods require that the transition between regions of different mesh fineness be treated separately, with an ad hoc form of the difference equations, thus increasing the size of the program and possibly lowering the precision of the calculation. Generally these methods do not lend themselves as easily as the present one to dynamic mesh adjustment.

The linked data structure and the two basic addition and deletion procedures create an extremely flexible computing environment, allowing fast dynamic modifications of the local mesh fineness according to the calculation needs. The calculation itself proceeds even faster than it would on a two-dimensional array representing a square grid with the same number of points, because of the faster access time of the linked data structure; this is obtained at the expense of the additional memory space necessary to store the links.

3. Difference Formulation of the Navier–Stokes Equations

The dynamically adjustable data structure described in the previous section was conceived with an explicit calculation scheme in mind. This choice was dictated by the difficulty of devising a time-efficient implicit calculation scheme over a variable-spacing mesh devoid of coordinate lines, and by the speed improvements obtainable for the explicit method in stationary problems, where a "false transient" not representing the actual time evolution of the problem can be used, with an equivalent "time step" adjusted for each point at the stability limit. Special care is needed only in the treatment of first derivatives, in order to insure stability while preserving central-difference accuracy.

The chosen differential formulation of the incompressible Navier–Stokes equations over each cross is the standard stream function-vorticity formulation [10], which automatically reduces to the equation $\Delta_2 \psi = 0$ in the irrotational flow regions. In connection with the present method this formulation has the additional advantage that, being ψ and ω scalars, the relevant equations are invariant to rotation, contrarily to what happens to the separate components of a vectorial equation.

The dimensionless equations are therefore written in a generic cartesian coordinate system as

$$\Delta_2 \psi = \omega; \tag{1}$$

$$\Delta_2 \omega = R(\psi_y \omega_x - \psi_x \omega_y) - \omega^+, \tag{2}$$

where ψ denotes the stream function, ω the vorticity, R the Reynolds number, and ω^+ the vorticity production, i.e., the curl of external volume forces.

The transport equation for a generic fluid property f (temperature, concentration of some component, etc.) can be added to Eqs. (1)–(2) and is

$$\Delta_2 f = N_f R(\psi_y f_x - \psi_x f_y) - f^+. \tag{3}$$

N_f is the dimensionless transport number pertaining to the property f (Prandtl number, Schmidt number , etc.) and f^+ is the production of f per unit volume.

The difference formulation of Eqs. (1)–(3) over a cross adopting central differences for all first derivatives does not lead to a simple and stable explicit solution method. In fact, extracting the value of each unknown at the cross center from the corresponding equation and recalculating it at each step is known to be a stable method only when the cell Reynolds number is low enough, which is not true in many interesting cases. The standard manner to recover stability by adopting upwinded, rather than central, first differences, has second-order accuracy lost.

Our solution to the stability problem is a variant of the upstream-weighted differencing scheme [11, 12], modified so as to give the same final accuracy as central differencing together with the stability properties of upstream differencing.

The idea of the method is to use the upwinded form of the equations, but modify each convective term through a suitable correction factor in order to restore the approximation of central differences. Assuming $\psi_u - \psi_d > 0$ and $\psi_r - \psi_l > 0$, so that the proper upwinded differences are $f_c - f_l$ and $f_u - f_c$, the difference approximation of Eq. (3) is written as

$$\begin{aligned} & f_r + f_u + f_l + f_d - 4f_c \\ & \quad = N_f R[C_1(\psi_u - \psi_d)(f_c - f_l) - C_2(\psi_r - \psi_l)(f_u - f_c))]/4 - h^2 f^+, \end{aligned} \tag{4}$$

$$C_1 = (f_r - f_l)/(f_c - f_l); \qquad C_2 = (f_u - f_d)/(f_u - f_c) \tag{5}$$

and the equations for ψ and ω can be obtained as particular cases by letting, respectively, $R = 0$ and $f^+ = -\omega$ or $N_f = 1$ and $f^+ = \omega^+$. The subscripts r, u, l, d, c denote the right, upper, left, lower vertices, and the center of the cross; h is the distance of any vertex from the center. If $\psi_u - \psi_d$ and/or $\psi_r - \psi_l$ are negative, the upwinded differences must be taken in the opposite direction in both Eqs. (4) and (5).

Equation (4) is a central difference equation and yields second-order accuracy. In the iteration process, however, Eq. (4) is treated as an upwinded equation, and f_c is recalculated at each step as

$$\begin{aligned} f_c = & \{f_r + f_u + f_l + f_d - af_c + N_f R[C_1(\psi_u - \psi_d)f_l \\ & + C_2(\psi_r - \psi_l)f_u]/4 - h^2 f^+\}/\{4 - a + [C_1(\psi_u - \psi_d) \\ & + C_2(\psi_r - \psi_l)]/4\}, \end{aligned} \tag{6}$$

while the old value of f_c is used inside C_1 and C_2.

The parameter a can be used to overrelax (if positive) or underrelax (if negative) the iteration process. A certain amount of overrelaxation increases the convergence rate of the iteration, but makes it unstable unless additional modifications are introduced; underrelaxation can be necessary, even if it slows down the calculation, to maintain stability when motion and transport equations are coupled.

For motion equations alone, Eq. (6) with $a = 0$ can be seen to restore the stability that was lost by the adoption of central differences, with the only exception discussed below, and while it has the same convergence properties as an upwinded-difference scheme, its limiting steady state is the solution of Eq. (4), which is second-order accurate.

By experimenting with Eq. (6) and observing the evolution of the variables under this iteration scheme, one can notice a definite smoothing out of perturbations and convergence to a steady solution, except for some well-delimited vortices, which suddenly appear in the part of the field where the vorticity is expected to become negligible and then decay slowly while being transported with the stream. These vortices are due to the presence of a denominator in Eqs. (5), whose value in a region where f becomes very small is determined mostly by numerical errors and can vanish at random times. Consequently the correction coefficients of Eqs. (5) can undergo wild fluctuations, with the observed sudden appearance of unexpected vortices. The remedy to this situation is to check the value of the correction coefficients at each step and restrict them within definite bounds. The exact choice of these bounds is not very critical; in fact wherever the approximation of derivatives by finite differences is meaningful the ratio between the central and upwinded differences must be close to 2; where this ratio is much different from 2 the approximation is bad anyhow and altering the value of the coefficient cannot make it any worse. From a different point of view, we are considering a region where the solution of the differential equations is of very small magnitude and the solution of the finite equations consists mainly of noise; provided the noise is not amplified, i.e., the algorithm is stable, it is not important whether the equations are altered in this region.

Given the above considerations, one can arbitrarily choose to restrict the correction coefficients between, say, 1 and 3. A less arbitrary choice was also explored, but the difference in the results is minimal. With the addition of coefficient range checking a simple algorithm was obtained which proved accurate and stable in all of our tests.

A further improvement in the speed of this algorithm was obtained by a careful choice of the recalculation order of the grid points. Since the crosses are maintained in a linked list, no ordering is privileged as far as access time is concerned.

Three orderings were tested:

The first one is generated by the routine used to decrease the size of a cross when needed, inserting any required additional crosses in the list immediately after the cross whose reduction caused the addition: therefore, in this case, crosses that are close in the list tend to be also spatially close to each other.

The second ordering is also generated by the cross-reduction routine inserting each additional cross at the top of the list in the moment it is created. Contrary to the first case, now points which are immediate neighbours tend not to be close in the list.

The third ordering is substantially different, because it is changed at each iteration step, using a criterion meant to create a streamwise arrangement of crosses in the list. For this purpose, during each step the list is scanned in top-down order and the crosses are flagged as they are recalculated, checking also whether or not the downstream crosses have been already recalculated in the same step. If they have been, the cross is moved to the top of the list. This method, which requires very little calculation overhead, proves beneficial to stability, even if no fixed ordering is ever definitely established in the list.

In fact, while in several test runs using the first two orderings no meaningful difference was observed, the third method was found to allow a certain degree of overrelaxation without losing stability: Eq. (6) can be used with $a = 1$ in connection with the third ordering, whereas $a \leqslant 0$ is mandatory in connection with the other two orderings.

Finally, we want to point out that, although we have observed the stability of Eq. (6) in several cases, stability cannot be always guaranteed. In particular, when other unknowns besides ψ and ω are present, as, for instance, in convection problems, oscillations of steady amplitude, very small in some cases, have been observed, caused by the coupling between the equations. This situation, as well as a way to cure it, is further discussed in Section 7.

4. Boundary Conditions

At the boundary of the solution domain there exist incomplete crosses, that is, crosses whose arms pointing outwards do not correspond to any mesh point. These are the points were the boundary conditions appropriate to the problem at hand must be imposed.

In the problem represented by Eqs. (1)–(3), generally, a condition for each equation will be needed along all the boundaries. We shall assume a square or rectangular boundary such that the centers of the incomplete crosses fall exactly on it and we shall not be concerned in the following with boundaries passing between mesh points. If necessary, this kind of boundary can be treated by our method using suitable interpolations or combining the adaptive-grid method with a fixed coordinate transformation mapping the curved boundary onto a square one. If a conformal mapping is adopted, mixed second derivatives are not introduced by the transformation and the validity of the present method is preserved.

For rectangular boundaries the formulation of boundary conditions parallels closely the standard one used for uniform square grids, the only difference being the coexistence along the boundary of straight-up and tilted crosses. In particular the no-slip condition along a rigid wall, expressed in the $\psi - \omega$ formulation by the Thom technique [13] is, in the case of a straight-up cross where for instance the non-blank link is "up,"

$$\omega_c = 2(\psi_u - \psi_c)/h^2, \tag{7}$$

whereas for a tilted cross it is

$$\omega_c = 2(\psi_u + \psi_r - 2\psi_c)/h^2. \tag{8}$$

In a similar way, a three-point no-slip condition can be also formulated. It was tested by us and did not show in connection with the present method a very significant improvement over the two-point condition. Results of this test are reported in Section 6.

Along with the no-slip condition, a condition on each additional f must be given. When f itself is assigned, the condition is immediate; on the other hand, when the condition is given on the normal derivative of f (or on a linear combination of f and its normal derivative), second-order accuracy may be lost if the simplest expressions are used, i.e.,

$$f'_c = (f_u - f_c)/h \tag{9}$$

for a straight-up cross, and

$$f'_c = (f_u + f_r - 2f_c)\ /2h \tag{10}$$

for a tilted cross. A second-order accurate expression of the normal derivative can be obtained also in this case by a 3-point formula, which was tested by us and did not yield a really significant improvement over the results obtained with Eqs. (9)–(10).

A completely different kind of boundary conditions arise in connection with infinite domains [14]. This is a complex subject in itself that we can only touch superficially here; however, a few comments are in order. In dealing with infinite domains two alternatives are possible: to use a transformation of variables to map the infinite domain onto a finite one, or to cut the infinite domain along some far enough line and substitute the remaining part with boundary conditions along that line. The two possibilities are not as different as they may appear at first sight: after the change of variables the infinity of the original domain is mapped onto a singularity in the transformed domain, and a knowledge of the behaviour of the solution in a neighbourhood of the singularity is necessary in order to formulate correctly the numerical problem. Finding this behaviour amounts to the same problem as finding the appropriate boundary conditions along a far line in the original domain. On the other hand, when the original domain is retained, generally a non-uniform mesh is required which corresponds to the use of a uniform mesh in transformed coordinates.

Our self-adapting method is particularly well-suited to solve these problems in the original infinite domain, since the crosses can automatically grow as large as required in approaching the far truncation line, but the appropriate boundary conditions must be found almost on a problem-by-problem basis.

5. Dynamic Mesh Adjustment

The mesh structure described in Section 2 can be easily manipulated in order to make it finer where required and coarser where sufficient.

The manipulation can be done by an operator interactively or by an automatic procedure built into the computer code which, periodically after a fixed number of iterations of the calculation, verifies the adequacy of the mesh and modifies it accordingly. The key step in writing such a procedure is to define an adequacy function which the routine can apply to decide whether the size of the crosses in any given area needs to be changed. The idea of an automatic process using such a function is not new [1, 2]. It is agreed that, for lack of a more direct estimation of the truncation error of the difference approximation, the decision must be based on the order of magnitude of the derivatives of the unknown functions, adopting a more closely spaced mesh where the derivatives are higher, and thus the variations sharper, and a coarser mesh where the derivatives are lower. However, while the qualitative significance of this criterion is clear, its practical application is not problem-free, the main difficulty being that one would like to measure the "order of magnitude" of the derivatives in a given region, rather than their value at a single point. For instance, the casual vanishing of a particular derivative does not mean that an infinite step is adequate. The solution cannot be to take an average of the derivatives over a small region, because the determination of the correct size of this region can only be based upon an understanding of the characteristics of the solution which can hardly be embodied in a computer program.

This being the situation, we could not find a single criterion enabling the computer to find a good mesh for all possible problems. This would be a rather ambitious task, but we did find several criteria, each of which is best-suited to a different range of problems.

The first and simplest criterion tests the second derivatives of all the unknowns at each point, through their finite-difference approximations, imposing the condition that

$$h^2 < r \min_f (f_{\text{typ}}/f''_{\text{M}}), \tag{11}$$

where h is the cross size, f''_{M} is the maximum absolute value of the second derivatives of f taken along the two directions of the cross arms, f_{typ} is a typical value of f and $\min_f$ denotes the minimum taken over all the unknowns, ψ included. r is a factor which must be assigned by the operator based on the precision he wants to obtain. The computer must check that h be the largest possible value compatible with Eq. (11) and modify the mesh accordingly.

In this method the danger of a very large step size being chosen because of the vanishing of some derivative is probabilistically defeated by the simultaneous testing of several quantities. On the other hand, Eq. (11) contains "typical" values which must be arbitrarily assigned and are the same for the whole field, while in problems where a small region with a peculiar behaviour such as a boundary layer exists, different typical values in different regions would be more appropriate. Because of the globally assigned typical value of each unknown f we call this a "global" criterion; the other methods we are going to describe tend mainly to correct the choice of typical values and to make it independent of the judgement of the operator. As a second, minor, fault, Eq. (11) is not completely isotropic, that is, equally fair to all directions of space, because f''_M takes into account only the second derivatives along two orthogonal directions. (It is not possible to give a finite-difference approximation of a mixed second derivative using only five points forming a cross.) This defect is implicit in the cross scheme and is common to the other criteria; it does not seem to cause practical problems.

In spite of its disadvantages, the global criterion works well, at least when boundary layers are not exceedingly pronounced, and is simple.

To derive an estimation of the correct step size without relying on typical values, it is possible to use both the first and second derivatives of each unknown, together with the value of the function itself, to form a quantity with the dimensions of a length to which the cross size can be compared. Our second criterion is then

$$h < r \min_f \left[\max(f'_M/f''_M, |f|/f'_M)\right], \tag{12}$$

where again f''_M is the maximum over the two possible directions of the absolute value of the second derivative and f'_M of the first derivative. f'_M/f''_M and $|f|/f'_M$ must be simultaneously tested to avoid that a null size h be forced where either $|f|$ or f'_M is vanishing.

This "local" criterion works better than the global one in conditions where large inhomogeneities exist in the solution field; however, in fluid-dynamic problems the stream function ψ must be excluded from the test; in fact, the simultaneous vanishing of ψ and its derivatives, which would be unlikely to happen by chance, is purposely imposed as a boundary condition on any rigid wall. Consequently, if the test of Eq. (12) is extended to ψ as well as to the other variables, the program tends to refine indefinitely the mesh in proximity of the wall. On the other hand, when ψ is excluded the program may choose too big a mesh size in the irrotational parts of the flow, where ψ is the only non-zero quantity, and an upper bound must be imposed on size to take care of these regions.

But, the worst defect of Eq. (12) is that it overestimates the fineness required at the outer edge of a boundary layer. In fact, at the outer edge of a boundary layer, vorticity typically tends towards 0 as e^{-ay^2}, y being the normal coordinate to the boundary and a an appropriate coefficient, so that $|\omega_y|/|\omega|$ and $|\omega_{yy}|/|\omega_y|$ both tend to increase linearly with y. Consistently, the program using Eq. (12) chooses smaller and smaller spacings going from the boundary layer out into the irrotational outer region, up to the point where the exponential decrease of vorticity is overridden by numerical errors.

While both the global and the local criteria are workable, the global one suffers by the use of a single "typical" value, not being able to deal properly with a region such as a boundary layer where different "typical" values would be needed, and the local criterion suffers by trying to represent with exceeding accuracy a region of exponential decay, not being able to recognize the scarce importance of the small values existing in this region with respect to the global situation. While it is clear that a neither completely global nor completely local criterion is needed, we already pointed out the difficulty of devising such a criterion in general. However, for the specific cases where boundary layers are clearly identifiable in the solution domain, a useful criterion is one similar to the global criterion which uses as a typical value of the vorticity the maximum of the absolute value of this function along the line connecting the test point to the closest wall. (In addition a global typical value of the stream function must still be assigned.) In a situation where a boundary layer exists close to each wall, this criterion has the effect that a typical value characteristic of the local section of the boundary layer is chosen, and in such a situation it overcomes all the defects of both the global and the local criteria.

This third, semi-local, criterion worked best, among the tested ones, in the problems where a clear boundary layer structure exists. If, however, this structure is not well-defined one of the other two criteria is preferable.

6. Comparison with an Exact Solution

To show the efficacy of the described method we shall present the solution of two test problems: the Hiemenz stagnation flow, for which an exact solution of the Navier–Stokes equations is known [15], and the free-convection flow in a square cavity. The latter problem is the topic of the next section.

The Hiemenz flow is generated by a stream impinging normally onto an infinite plane wall, where a two-dimensional stagnation point is formed (Fig. 2).

The corresponding solution of the Navier–Stokes equations in $\psi-\omega$ form is $\psi = xF(y)$, $\omega = xF''(y)$, where F must satisfy the equation $F''' + FF'' - F'^2 + 1 = 0$, with the conditions $F(0) = F'(0) = 0$, $F'(\infty) = 1$. This test case was chosen, among those for which an exact solution is known, because of the requirement that a boundary layer should appear in it, in order to exploit the automatic mesh fineness adjustment, notwithstanding the numerical complications deriving from the infinite definition domain of the solution. In order to test our variable spacing method free

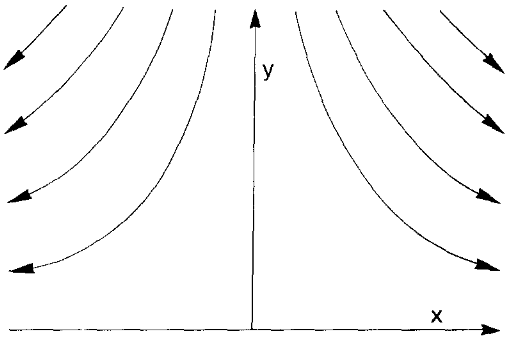

FIG. 2. Hiemenz stagnation flow.

from the errors caused by the truncation of the infinite domain, we rely on the knowledge of the exact solution for choosing the appropriate truncation conditions.

For the numerical solution of the Hiemenz-flow problem, we adopt a square domain with a vertex in the stagnation point and two sides lying along the x- and y-axes. The boundary conditions on the left side of this square, which is a symmetry axis of the solution, are $\psi = 0$, $\omega = 0$. On the lower side, which is a rigid wall, $\psi = 0$ and $\psi_y = 0$. (The latter can be transformed into a condition for ω by using either a two-point or a three-point Thom formula. Both have been tried, with the results reported below.) On the upper side the flow field is given by the asymptotic irrotational form of the stream coming from infinity, i.e., $\psi_y = x$, $\omega = 0$. Of course the size of the square must be chosen large enough for this approximation to be valid, but since its error is of exponentially small order this condition is not very critical. Contrarily, on the right side of the square the boundary conditions *are* critical, because the boundary crosses both an irrotational and a boundary-layer region. However, for this particular problem, knowing the form of the exact solution, it is possible to introduce an exact boundary condition in the form of a relation between ψ, ω and their normal derivatives

$$\psi_{\mathrm{n}} = \psi/L, \qquad \omega_{\mathrm{n}} = \omega/L,$$

L being the side of the square domain. These relations are exactly verified for any L by the solution, which depends linearly on x, and therefore no error is introduced because of this condition.

In the following tests, the equations were discretized using Eq. (6) with $a = 1$. The adjustment criterion was the third one presented in Section 5, with regard to ω only, and a maximum size was imposed on the crosses to keep them from growing too large in the irrotational region.

Tests were run for different values of the fineness parameter r and the square side, and for two different discretized no-slip conditions. The imposed maximum cross size was 1 when $r \leqslant 0.1$ and was $\frac{1}{4}$ the square side otherwise. Each case was terminated when the maximum modulus of the difference between successive iterates of ψ was less that 5×10^{-5}, and the result has been used as the starting point for the following case. (If the calculation is started directly with a low value of r instead of decreasing it gradually the total time employed is considerably longer.) In Table I are reported, for a square side of 6.4 and using the two-point boundary condition, the number of steps required to complete the calculation, the number of points present in the mesh at the end, the time employed on our machine (HP 9826 desktop computer) and the value of the vorticity at the center of the horizontal wall divided by x with its relative error (the exact value is 1.23259), for several values of the fineness parameter r.

The number reported in the "point-steps" column is the number of single-point recalculations performed, which is not the product of the number of points and the number of steps because the number of points is variable. The number of point-

TABLE I

Hiemenz Flow in a Square with Side 6.4 Using the Two-Point No-Slip Condition

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	ω_w/x ($x = 3.2$)	Error
—	5×5	86	8	2150	0.9268	−25%
0.8	IDEM					
0.4	71	184	38	7927	1.0905	−12%
0.2	129	328	139	25831	1.2523	+1.6%
0.1	157	428	234	41588	1.2083	−2.0%
0.05	334	601	599	98627	1.2349	+0.19%
0.02	493	723	1006	159074	1.2285	−0.33%

TABLE II

Hiemenz Flow in a Square with Side 12.8 Using the Two-Point No-Slip Condition

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	ω_w/x ($x = 6.4$)	Error
—	5×5	100	9	2500	0.5783	−53%
0.8	50	177	27	6245	0.9053	−27%
0.4	106	284	88	17301	1.0738	−13%
0.2	210	527	392	67005	1.2374	+0.39%
0.1	443	824	1216	195020	1.2121	−1.7%
0.05	732	1212	3099	477792	1.2327	+0.01%

TABLE III

Hiemenz Flow in a Square with Side 6.4 Using the Three-Point No-Slip Condition

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	ω_w/x ($x = 3.2$)	Error
—	5×5	76	7	1900	1.2760	+3.5%
0.8	IDEM					
0.4	50	145	23	5059	1.1799	−4.3%
0.2	98	236	74	13683	1.2613	+2.3%
0.1	157	362	196	33727	1.2418	+0.75%
0.05	329	545	594	93938	1.2483	+1.3%

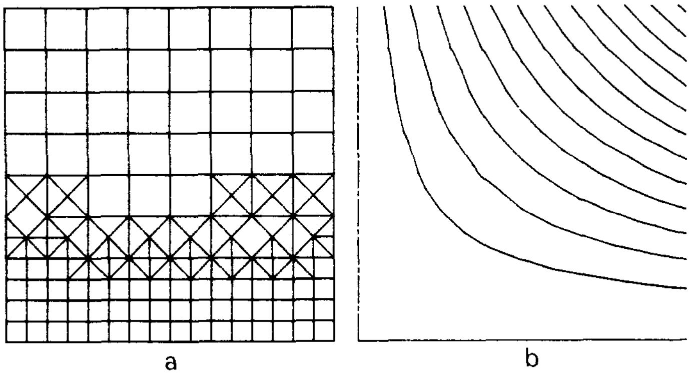

FIG. 3. Pattern of crosses generated in the solution of the Hiemenz flow problem (a) and resulting streamlines (b).

steps is directly proportional to the calculation time and can be useful to estimate the calculation time on other machines.

Table II reports similar information for a square side of 12.8, still using the two point boundary condition, and Table III again for a square side of 6.4, but with the three-point boundary condition.

Table I shows that the number of points in the mesh N is roughly proportional to r^{-1}, as can be expected, noticing that Eq. (12) makes the cross size h about proportional to $r^{1/2}$ and the number of points is proportional to h^{-2}. The relative error of the wall vorticity (skin friction) appears to decrease approximately as r^2 and for $r=0.1$ is of the order of 1%. That an error of this order of magnitude can be obtained for $r=0.1$ was generally observed in the majority of the other tests.

As can be seen from Table II, the results obtained for a square side of 12.8 are not significantly different from the ones reported in Table I; whence one can deduce that a square side of 6.4 is large enough. Also the results reported in Table III do not differ significantly from the previous ones, showing that, in this context, the use of a three-point no-slip boundary condition does not afford an improvement over the simpler two-point condition.

The pattern of crosses generated by the program for the case $r=0.1$ of Table I is reported as an example in Fig. 3a. The cross size is smaller in the boundary layer region and increases gradually outwards. It is worth noting that the boundary layer thickness for Hiemenz flow is constant. It also can be seen clearly in Fig. 3a how the cross-based method takes care automatically of the transition between small spacing and large spacing regions.

The well known streamline pattern resulting from the calculation is shown in Fig. 3b.

7. Natural Convection in a Square Cavity

The natural convection in a square cavity whose vertical walls are held at different fixed temperatures and whose horizontal walls are adiabatic (double glazing problem, Fig. 4) has been used by several authors [16, 17] as a test problem for numerical methods of solution of the steady Navier–Stokes equations with transport. In particular Ref. [16] contains a uniform-mesh implicit solution of this problem, which can be usefully compared with the one obtained by the present adaptive technique. (A uniform-mesh explicit solution on the same 65×65 grid as used in Ref. [16] cannot be obtained in a reasonable time.)

For natural convection , in the Boussinesq approximation, Eqs. (2)–(3) must be specialized as

$$\Delta_2 \omega = (\psi_y \omega_x - \psi_x \omega_y) - GT_x \tag{13}$$

$$\Delta_2 T = P(\psi_y T_x - \psi_x T_y), \tag{14}$$

where T is the dimensionless temperature increase, G is the Grashof number and P is the Prandtl number. The reference velocity has been chosen so as to make $R = 1$, and GT_x is the vorticity production due to gravitational forces.

The boundary conditions for double glazing are $\psi = \psi_n = 0$ on the four walls, $T(0, y) = 0$, $T(1, y) = 1$, $T_y(x, 0) = T_y(x, 1) = 0$.

To obtain the discretized formulation of this problem, Eq. (6) must be written three times, for ψ, ω and T, respectively. In particular, in the ω-equation the production term ω^+ is not zero and, since it is not invariant to reference frame

ferences have been used for this term in both cases, obtaining for straight-up crosses

$$h^2 \omega^+ = Gh(T_r - T_l)/2 \tag{15}$$

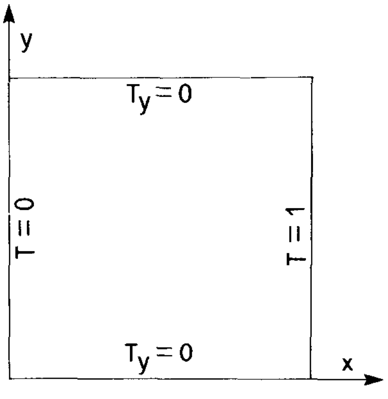

FIG. 4. Cavity convection (double glazing) problem.

and for tilted crosses

$$h^2\omega^+ = Gh(T_r + T_d - T_l - T_u)/2^{3/2}. \tag{16}$$

Just because of the coupling between the ω and T equations caused by the buoyancy term, when the Grashof number is high the iteration scheme based on Eq. (6) with $a = 1$ does not lead to a steady solution, but rather to an oscillation of steady amplitude, even though in many cases the oscillation is small and does not prevent an acceptable result from appearing. This oscillation, however, makes it difficult to terminate the calculation by an automatic criterion such as to check the magnitude of the change of variables between successive iterates. A similar behaviour was also noticed in Ref. [16] using an implicit algorithm, and it was already pointed out that it can be eliminated by underrelaxing the ω-equation with respect to the T-equation.

To explore the origin of this instability one can observe that if the vorticity production depended explicitly on ω, for instance being $\omega^+ = A\omega$, and instability would arise unless we made $a \leqslant -Ah^2$. In Eqs. (13)–(14) a constant such as A does not exist, but the vorticity production depends implicitly on ω through the temperature equation, and the coupling grows stronger with the coefficient Gh which appears in the discretized form of the buoyancy term, Eqs. (15)–(16).

As a consequence we assumed that in our scheme the oscillation can be eliminated by choosing, only in the ω-equation, a relaxation parameter a equal to $1\text{-}kGh$, and determined empirically k as 5×10^{-4}, for a Prandtl number of order unity. This modification allows a solution to be obtained for a Grashof number at least as high as 10^7, even though convergence is slow.

The double glazing problem has been solved by the present variable spacing method for $P = 0.71$ and the same values of the Grashof number considered in Ref. [16], adding the case $G = 1.40845 \times 10^7$ not considered there. The global adjustment criterion of Section 5 has been used, with the maximum absolute value of each quantity (ψ, ω, T) as its typical value. The conditions $\psi_n = 0$ and $T_n = 0$ have been imposed using either two-point or three-point difference formulas.

Each case has been run for several values of the fineness parameter r, arresting the calculation when the maximum modulus of the difference between successive iterates of the temperature was less than 10^{-6} for $G < 10^6$ and 10^{-5} for the last two cases.

As in the previous test, a significant difference was not observed between the results obtained using either two-point or three-point formulas for the boundary conditions.

The results obtained are reported in Tables IV–VIII.

In all these cases the number of points in the mesh, N, is found again to vary as r^{-1}; moreover it can be noticed again that making $r = 0.1$ is sufficient to give an error of the order of 1 %. This is particularly interesting in view of the wide range of Grashof numbers covered. In fact, as G increases the fluid-dynamic field changes substantially and boundary layers develop; nevertheless the automatic mesh-

TABLE IV

Cavity Convection for $G = 1.4084 \times 10^3$

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	Nusselt no.	Error
—	5×5	73	10	1825	1.1639	+4.1%
0.8	69	151	54	7176	1.1410	+2.0%
0.4	131	289	227	24684	1.1364	+1.6%
0.2	231	472	686	66821	1.1252	+0.60%
0.1	499	1049	4184	357229	1.1182	−0.03%
0.02	2261	2121	23733	2183241	1.1185	—
From Ref. [16] (65×65 uniform mesh):					1.118	
From Ref [18] (33×33 modes):					1.1178	

TABLE V

Cavity Convection for $G = 1.4084 \times 10^4$

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	Nusselt no.	Error
—	5×5	63	8	1575	2.2581	−0.18%
0.8	91	197	125	13489	2.3384	+3.4%
0.4	160	353	381	38403	2.3639	+4.5%
0.2	322	493	886	83190	2.2897	+1.2%
0.1	562	891	3538	305841	2.2622	—
From Ref. [16] (65×65 uniform mesh):					2.250	
From Ref. [18] (33×33 modes):					2.245	

TABLE VI

Cavity Convection for $G = 1.4084 \times 10^5$

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	Nusselt no.	Error
—	5×5	82	11	2050	2.8509	−37%
0.8	157	412	540	52103	5.0537	+11.3%
0.4	311	701	1557	141390	4.6758	+3.0%
0.2	523	1077	3914	339236	4.6318	+2.0%
0.1	1054	2097	17340	1413139	4.5749	+0.76%
0.05	1992	2845	37002	2907352	4.5405	—
From Ref. [16]: 65×65 uniform mesh:					4.573	
26×26 stretched mesh:					4.595	
From Ref. [18] (65×65 modes):					4.523	

TABLE VII

Cavity Convection for $G = 1.4084 \times 10^6$

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	Nusselt no.	Error
—	5×5	77	12	1925	2.5851	−70%
0.8	268	262	520	45897	8.5766	−1.7%
0.4	493	551	2200	187299	8.5883	−1.6%
0.2	935	918	6551	530816	8.7069	−0.2%
0.1	1871	1031	9275	737387	8.7241	—
From Ref. [16]: 65×65 uniform mesh:					9.272	
26×26 stretched mesh:					9.066	
From Ref. [18] (65×65 modes):					8.826	

adjustment procedure chooses the right number and position of the points, giving for each case an error of the same order of magnitude.

At the bottom of each table the values of the Nusselt number calculated in Ref. [16] are reported. A general agreement is observed; although the difference is not eye-catching, it appears that the results of Ref. [16] lose precision at the higher Grashof numbers, while ours maintain their precision because of the possibility of accurately following the boundary layers.

It must also be said that the implicit algorithm of Ref. [16] is faster. However, its use is bound to a certain kind of mesh, while our self-adapting mesh allows problems with a higher Grashof number to be solved at all, if slowly.

The isothermal lines and the streamlines corresponding to the results of Tables IV–VIII are plotted in Figs. 5–9, using the same representation as in Ref. [16]. At low Grashof numbers our results and those of Ref. [16] are practically coincident, but with growing G a slight discrepancy arises because of the better ability of our adaptive mesh to model the boundary layers.

It is also interesting to note how at high G the motion tends to be confined to the

TABLE VIII

Cavity Convection for $G = 1.4084 \times 10^7$

Fineness (r)	No. of points	Step no.	Time (sec)	Point-steps	Nusselt no.	Error
—	5×5	175	27	4375	2.4280	−84%
0.8	465	864	3704	311324	15.4579	−0.38%
0.4	853	1485	10134	839541	15.5162	—
From Ref. [18] (65×65 modes):					16.51	

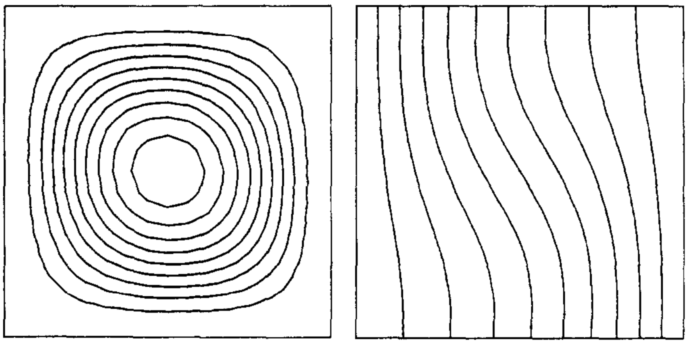
FIG. 5. Streamlines (left) and isothermal lines (right) for $G = 1.4084 \times 10^3$. ψ increment is 0.169, T increment is 0.1.

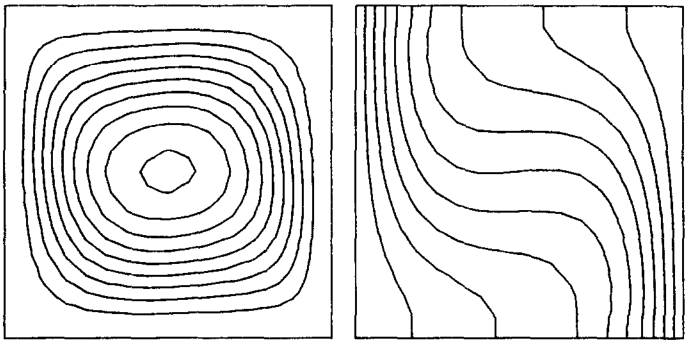
FIG. 6. Streamlines (left) and isothermal lines (right) for $G = 1.4084 \times 10^4$. ψ increment is 0.777, T increment is 0.1.

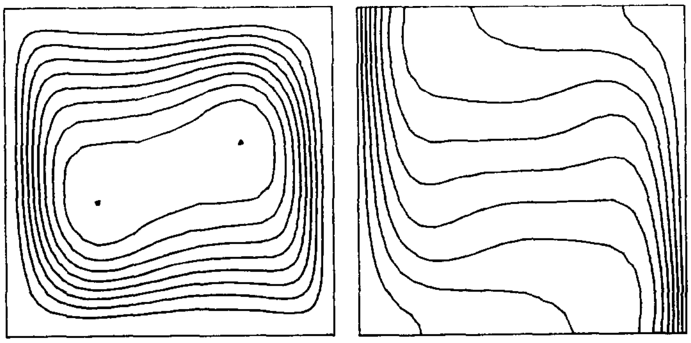
FIG. 7. Streamlines (left) and isothermal lines (right) for $G = 1.4084 \times 10^5$. ψ increment is 1.493, T increment is 0.1.

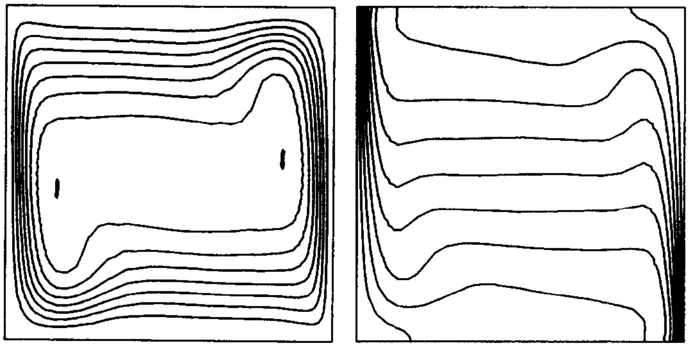
FIG. 8. Streamlines (left) and isothermal lines (right) for $G = 1.4084 \times 10^6$. ψ increment is 2.958, T increment is 0.1.

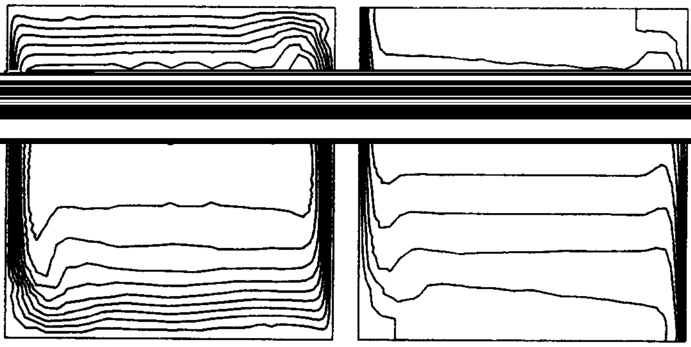
FIG. 9. Streamlines (left) and isothermal lines (right) for $G = 1.4084 \times 10^7$. ψ increment is 4.5, T increment is 0.1.

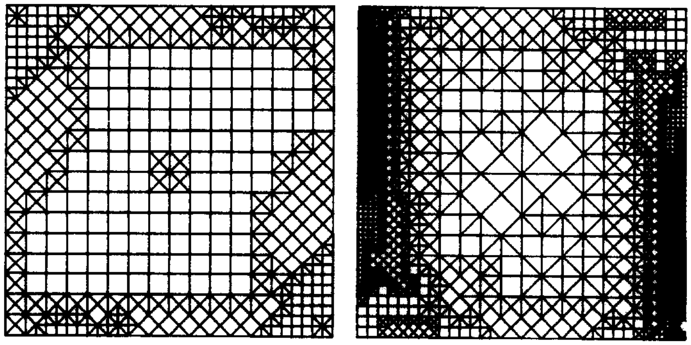
FIG. 10. Pattern of crosses generated for the cavity convection problem for $G = 1.4084 \times 10^3$ (left) and $G = 1.4084 \times 10^6$ at fineness parameter $r = 0.1$.

boundary layers existing near the walls, while in the central region the fluid is relatively still, with a stable, vertically stratified temperature distribution.

The arrangement of crosses generated by the computer program is shown in Fig. 10 for the lowest and the highest considered G, at a fineness parameter $r = 0.1$. It appears immediately that the grid corresponding to the higher G contains many more crosses than the other one, and most crosses tend to gather along the vertical walls, and in particular near the upper-left and lower-right corners where the boundary layers are thinnest. We want to stress again that these two different grids, each created for its particular problem, yield an error of the same order of magnitude.

8. Conclusions

An adjustable variable-spacing grid has been presented which allows single points to be added to or deleted from it at any time during the calculation. This grid is conceived so that each point is the center of a symmetrical cross formed with four other points of the mesh and therefore can be used for the numerical solution of two-dimensional, steady problems governed by partial differential equations in which second derivatives can be confined to the Laplacian operator. For this class of problems a second-order accurate difference approximation can be written over a cross-shaped cluster of points and used throughout the whole field of calculation. The resulting discrete equations are difficult to solve by an implicit method, but for steady problems an explicit, false-transient method proves fairly fast, provided that the adaption mechanism is operating and the equivalent time-step is adjusted at each point at the local stability limit.

An explicit solution method of the Navier–Stokes equations to be used over this grid has also been presented, adopting a hybrid form of the convective terms to obtain the accuracy of central differences together with the stability of upwinded differences.

Three adequacy functions which can be used to modify automatically the mesh during the calculation have been discussed.

A program based on these techniques has been tested in two cases: the Hiemenz stagnation flow, for which an exact solution is known, and the natural convection in a square cavity, for which other numerical solutions exist in the literature.

The conception of the method presented in this paper is founded on the separation of three sub-problems: mesh management, discretization of the equations and mesh adaption. For this purpose it was decided to create and delete points of the grid instead of moving them along, and the cross structure was developed to allow points to be added or deleted one at a time. In addition, having a variable number of points lets the computer choose the most suitable number for each problem.

Another advantage of our mesh-modification method over adaptive grid methods based on continuous coordinate transformations is that thickening of the mesh in a region does not imply unwanted widenings in other regions. A drawback is that implicit or ADI methods cannot be easily implemented, so that we decided to use an explicit solution method. However, in false-transient calculations, where only the final steady state is of interest, the explicit method was found to work much faster on the self-adapting grid than on an uniform grid with the same number of points, because of the use of large equivalent time-steps on the larger crosses together with small steps on the smaller crosses, pushing each at the local stability limit.

A critical point of all adaptive mesh methods is the error measure used to drive the mesh configuration, and although we found satisfactory measure definitions for particular problems, our experience confirms that a general criterion is difficult to determine. The modularity of our system, however, allows any function of the mesh variables to be defined as the error measure, and to change it only a single function definition inside the program must be changed.

In the test runs against an exact solution (Hiemenz flow), introducing a suitable boundary condition to handle the infinite domain, a good agreement was found, and no errors ascribable to the variable step size were observed.

For the cavity convection problem our results were compared with those of Ref. [16], obtained over a uniform mesh. A good agreement was found at the lower values of the Grashof number, for which no boundary layers exist in the flow, but at the higher Grashof numbers the uniform-grid method shows up unable to represent accurately the flow field, while the present variable-spacing method works well at least up to $G = 10^7$.

Our results are also in good agreement with those of Ref. [18], obtained by means of a spectral method in a time of the order of hours on an IBM 370/168.

Finally, an interesting feature of the cavity convection problem is that at high Grashof numbers the motion tends to be confined within a boundary layer near the walls, and in the center a calm region appears where the temperature is vertically stratified. Our variable-spacing method is appropriate for the solution of this kind of problems, where regions with completely different behaviour exist, especially when the position of these regions is not known in advance.

ACKNOWLEDGMENT

The author is indebted to A. Pozzi for his thorough reading of the manuscript and many useful suggestions.

REFERENCES

1. J. F. THOMPSON, *AIAA J.* **18**, 1505 (1984).
2. J. F. THOMPSON, Z. U. A. WARSI, AND C. W. MASTIN, *J. Comput. Phys.* **47**, 1 (1982).
3. J. F. THOMPSON, in *Numerical Grid Generation*, edited by J. F. Thompson (North-Holland, Amsterdam, 1982).
4. R. CAMARERO AND M. YOUNIS, *AIAA J.* **18**, 487 (1980).

5. P. D. Thomas and J. F. Middlecoff, *AIAA J.* **18**, 652 (1980).
6. B. L. Pierson and P. Kutler, *AIAA J.* **18**, 49 (1980).
7. H. A. Dwyer, R. J. Kee, and B. R. Sanders, *AIAA J.* **18**, 1205 (1980).
8. H. A. Dwyer, *AIAA J.* **22**, 1705 (1984).
9. J. P. Roache, *Computational Fluid Dynamics* (Hermosa, Albuquerque, N. M., 1976), Sect. III–A–1, Eq. (3.16).
10. J. P. Roache, *Computational Fluid Dynamics* (Hermosa, Albuquerque, N. M., 1976), Sect. II–B.
11. G. D. Raithby and K. E. Torrance, *Comput. Fluids* **2**, 191 (1974).
12. E. M. Parmentier and K. E. Torrance, *J. Comput. Phys.* **19**, 404 (1975).
13. J. P. Roache, *Computational Fluid Dynamics* (Hermosa, Albuquerque, N. M., 1976), Sect. III–C–2.
14. J. P. Roache, *Computational Fluid Dynamics* (Hermosa, Albuquerque, N. M., 1976), Sect. III–C–11.
15. H. Schlichting, *Boundary Layer Theory* (McGraw–Hill, New York, 1968), Sect. V–b–9.
16. T. N. Phyllips, *J. Comput. Phys.* **54**, 365 (1984).
17. I. P. Jones, *J. Fluid Mechanics* **95**, 4 (1979).
18. P. Le Quere and T. A. de Roquefort, *J. Comput. Phys.* **57**, 210 (1985).