JOURNAL OF COMPUTATIONAL PHYSICS 106, 52-61 (1993)

# Shepard's Interpolation for Solution-Adaptive Methods

CHERNG-YEU SHIEN* AND HELEN L. REED

*Mechanical and Aerospace Engineering Department, Arizona State University, Tempe, Arizona*

AND

THOMAS A. FOLEY

*Computer Science and Engineering Department, Arizona State University, Tempe, Arizona*

Received December 4, 1990; revised July 27, 1992

In numerical simulations of fluid-dynamics problems, solution-adaptive methods have proven to be very powerful. The implementation of the modified Shepard's interpolation to the structured grids used in CFD is suggested and described in this paper, which takes advantage of the logical grid structure. This technique, which is demonstrated to be robust, efficient, smoother, and a more accurate alternative to linear interpolation, is used in the remapping step during the solution-adaptation procedure. Applications to the solutions of the incompressible Navier Stokes equations (both 2D and 3D) are included. © 1993 Academic Press, Inc.

## 1. REVIEW OF PREVIOUS WORK

### 1.1. Introduction

In the past decade, solution-adaptive methods have proven to be very powerful tools in problems in computational fluid dynamics where large gradients exist. Compared to grid refinement, the moving-grid approach has been commonly used because of its easy implementation, especially for 3D flows. Saltzman and Brackbill [1] addressed the difficulties of multidimensional adaptive-grid generation through the variational approach. Subsequently, most methods have dealt with how to move the grid efficiently and how to choose the parameters to be specified in the adaptive-grid generation (e.g., Nakahashi and Deiwert [2]). Several good reviews on adaptive-grid methods are given by Anderson [3], Thompson [4], and Eiseman [5].

During the solution-adaption procedure, the (intermediate) solutions obtained from a PDE solver (e.g., a Navier–Stokes solver) must interact with the grid obtained from an adaptive-grid generator. The interaction occurs either through grid-speed terms in the PDE solver or simply by transferring the solution data from one computational grid to another, namely, remapping [5] (or rezoning). The advantage of the grid-speed approach is that a better resolution can be obtained in time, particularly, for the implicit schemes. But, since the grid-point locations are not known a priori, the grid-speed approach suffers from the difficulty to control coordinate singularity [5]. For example, if local grid speeds are too large or move too rapidly during the solution procedure, the result could be an overlapping or highly skewed grid. In contrast to the grid-speed approach, the remapping procedure is commonly used because of its simplicity, efficiency, and numerical stability [5], especially when steady-state solutions are the prime interest. It is also more practical for inherently time-consuming, unsteady, 3D flow calculations since the grid is adapted only occasionally during the computation. The main purpose of this paper is to consider how to apply the remapping step to solution-adaptive methods.

Recently, Mastin [6] presented some interpolation schemes to transfer solution data from one computational grid to another. One of his suggestions is to first use a point-search algorithm to find a grid point sufficiently close to $Q$, the point for which interpolated solution values are desired and then to determine a cell $C$ which contains $Q$, using a cell-search algorithm. Thus the value at $Q$ can be computed by using bilinear (2D) or trilinear (3D) interpolation since the solution values at the vertices of the cell $C$ are known. But this algorithm suffers from two difficulties. First, bilinear (2D) and trilinear (3D) interpolation cannot always be applied to arbitrary quadrilateral and hexahedral cells, respectively. Although Seldner and Westermann [7] have already given a generalization of how to apply bilinear interpolation to arbitrary convex quadrilateral cells, there is no corresponding generalization of trilinear interpolation yet (to the authors' knowledge). Second, the cell-search algorithm may fail to locate the point $Q$ within a cell $C$ in general 3D curvilinear coordinates if the faces of the cell are

* Present address: National Center for High-Performance Computing, Taiwan, R. O. C.

not planar. This is also pointed out by Mastin himself. Generally, this is a common case in solution-adaptive methods. Furthermore, the interpolant obtained from bilinear or trilinear interpolation is only a continuous function across the interfaces between cells. From the computational point of view, not only a continuous function, but rather a continuous, smooth function across the interfaces between cells would be preferred.

Because of the importance of the remapping step in solution-adaptive methods, an interpolation scheme is needed which is *robust*, *highly accurate* (at least a continuous, smooth function across the interfaces between cells), *efficient* (or at a reasonable computational cost), and *easy to implement*. Among different interpolation methods, the modified Shepard's interpolation [8–13] is an excellent candidate for solution-adaptive methods for the following reasons. First, it is designed for scattered data (it will be shown how to customize it to a well-structured grid later) so it is robust even when the faces of the cell are not planar. Second, the modified Shepard's method used in this paper has quadratic precision [12], which is more accurate than the bilinear and trilinear interpolation for 2D and 3D, respectively. It also satisfies the requirement that the resulting interpolant is a continuously differentiable smooth function across the interfaces between cells. Next, it is a local interpolation method which makes it efficient, with the level depending on how many points are involved in the calculation. Last, it is easy to implement for both 2D and 3D.

In the next section, the original Shepard's method and some of its properties will be described. Following this, the modified Shepard's method as given by Franke and Nielson [12] is presented.

## 1.2. Shepard's Method

Shepard [8] developed a technique for interpolating 2D scattered data and applied it to the fitting of geographic and demographic data. Most of the basic properties of Shepard's method can be understood readily through discussion of the 2D case. Also there is a straightforward extension from 2D to 3D for this particular method. Thus, only 2D interpolation is given in this section.

Let $f$ be a function with values $f_k$ at nodes $(x_k, y_k)$ for $k = 1, ..., N$, and define

$$F(x, y) = \sum_{k=1}^{N} W_k(x, y)\, Q_k(x, y) \Big/ \sum_{k=1}^{N} W_k(x, y). \quad (1.1)$$

For appropriate choices of $W_k(x, y)$ and $Q_k(x, y)$ that follow, $F$ is a bivariate interpolant with the property that $F(x_k, y_k) = f_k$, $k = 1, ..., N$. If $W_k = 1/d_k^\mu$ and $Q_k(x, y) = f_k$, then Eq. (1.1) can be written as

$$F(x, y) = \begin{cases} f_k, & (x, y) = (x_k, y_k) \text{ for some } k, \\ \left(\sum_{k=1}^{N} f_k/d_k^\mu\right) \Big/ \left(\sum_{k=1}^{N} 1/d_k^\mu\right), & \text{otherwise}, \end{cases} \quad (1.2)$$

where $\mu > 0$, and $d_k = [(x - x_k)^2 + (y - y_k)^2]^{1/2}$. It is easy to see from Eq. (1.2) that closer points make a larger contribution to the interpolant $F$. Note that Eq. (1.2) has exactly the same form as that found in Shepard [8].

Here, some of the properties of Shepard's method will be described briefly. More details may be found from Ref. [9–11]. Equation (1.2) may be rewritten in cardinal form as

$$F(x, y) = \sum_{k=1}^{N} f_k \bar{W}_k(x, y), \quad (1.3)$$

where

$$\bar{W}_k(x, y) = \begin{cases} \delta_{kj}, & (x, y) = (x_j, y_j) \text{ for some } j, \\ (1/d_k^\mu) \Big/ \left(\sum_{j=1}^{N} 1/d_j^\mu\right), & \text{otherwise}. \end{cases} \quad (1.4)$$

Multiplying Eq. (1.4) by $\prod_{j=1}^{N} d_j^\mu$, $\bar{W}_k(x, y)$ becomes

$$\bar{W}_k(x, y) = \prod_{\substack{j=1 \\ j \neq k}}^{N} d_j^\mu \Big/ \sum_{i=1}^{N} \prod_{\substack{j=1 \\ j \neq i}}^{N} d_j^\mu. \quad (1.5)$$

$\bar{W}_k(x, y)$ is a *continuous* function in the domain. This follows from the fact that the denominator of Eq. (1.5) never vanishes if the $(x_k, y_k)$ are distinct. In fact, if $\mu > 1$, then $\bar{W}_k$ is a $C^\infty$ function; thus all high-order derivatives of the interpolant $F$ are continuous for all $(x, y)$.

It is clear that except for the points $(x_k, y_k)$, the interpolant $F$ constructed by Shepard's method is *analytic* everywhere in the domain. Its shape depends on the values of $\mu$. For $0 < \mu \leqslant 1$, Shepard's interpolant is not a smooth function, for it has cusps or corners at the point $(x_k, y_k)$. For $\mu > 1$, the partial derivatives of $F$ at each point $(x_k, y_k)$ are zero, which produces a flat spot at the point $(x_k, y_k)$. Shepard [8] suggested $\mu = 2$. There are other interesting properties of Shepard's formula, a few of which are listed below:

a. It satisfies the maximum principle, i.e.,

$$\min_{1 \leqslant k \leqslant N} f_k \leqslant F(x, y) \leqslant \max_{1 \leqslant k \leqslant N} f_k. \quad (1.6)$$

b. $F$ is also a constant function if the values of $f$ are constant; i.e., if $f_k = C$, $k = 1, ..., N$, then $F = C$.

c. $F$ is a convex combination of $f_k$; this is based on the fact that $\sum_{k=1}^{N} \bar{W}_k(x, y) = 1$ and $\bar{W}_k(x, y) \geqslant 0$ for all $k$.

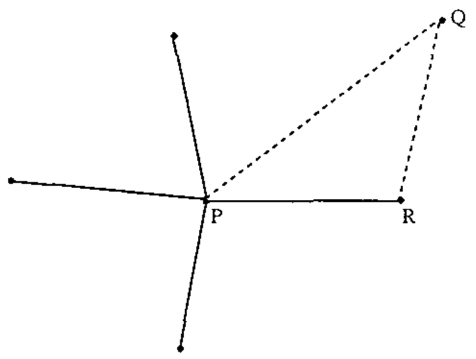

FIG. 2. Move from $P$ to $R$.

5. Define

$$\begin{aligned}Q_k(x,y) = {} & \bar{c}_{k1}(x-x_k)^2 + \bar{c}_{k2}(x-x_k)(y-y_k) \\ & + \bar{c}_{k3}(y-y_k)^2 + \bar{c}_{k4}(x-x_k) \\ & + \bar{c}_{k5}(y-y_k) + f_k\end{aligned}$$

and compute

$$F(x,y) = \sum_{k=1}^{N} W_k(x,y)\,Q_k(x,y) \bigg/ \sum_{k=1}^{N} W_k(x,y).$$

As pointed out in step 3 above, an efficient point-search algorithm is required for the proposed implementation. Recently, Mastin [6] presented interpolation schemes for solution-adaptive methods and related problems. He gave two search algorithms in his work, namely, the *point*-search algorithm and the *cell*-search algorithm, the former of which has been adopted in this study. To apply the point-search algorithm to find a grid point sufficiently close to $Q$, the point for which interpolated solution values are desired, assume $P$ is a starting point (see Fig. 2). First, both the distance from $P$ to $Q$ and the distance from $Q$ to $R$, where $R$ is a neighbor of $P$, are computed and compared with each other. $P$ is replaced by $R$ whenever $R$ is found to be closer to $Q$ than $P$ is. Then, the procedure is repeated until $P$ is closer to $Q$ than any of its four (2D) or six (3D) neighbors are.

## 3. ADAPTIVE-GRID SCHEME

Grid-generation codes based on the elliptic grid-generation equations are commonly used in computational fluid dynamics. Since the control-function approach is also based on these equations, it is natural to consider it among different adaption strategies [15, 16]. Additionally, it is easy to add the control adaptive functions (used in the control-function approach) to those already evaluated from the elliptic grid-generation codes in which the geometry is considered [15].

In order to improve the accuracy of the solutions, some sort of orthogonality control has to be added to the grid-generation equations used in the control-function approach. Here, the resulting Euler equations obtained by minimizing the integral of the orthogonality control, as proposed by Saltzman and Brackbill [1], are incorporated into the equations used in the control-function approach to improve the orthogonality of the interior grid [17, 18]. Also, the orthogonality of the grid at the boundary is enhanced by use of Neumann boundary conditions to solve the grid-generation equations.

## 4. NUMERICAL RESULTS

Driven-cavity flow has become a standard test case for the incompressible Navier–Stokes equations in the past two decades. For example, Ghia *et al.* [19] used a multi-grid method coupled with a strongly implicit procedure (SIP) to solve 2D driven-cavity flow up to $\mathrm{Re} = 10{,}000$ on a $257 \times 257$ grid. Schreiber and Keller [20] also solved the same problem on a $180 \times 180$ grid using Newton iteration coupled with a continuation method. Both used a uniform grid.

In the present work, the solution-adaptive method with modified Shepard's interpolation is used to obtain a solution yielding comparable results while using fewer grid points and computer resources. In this section, the problem formulation and solution algorithm for the 2D case are described briefly. Results obtained from 2D driven-cavity flow with $\mathrm{Re} = 5000$ on an $81 \times 81$ grid are shown. Finally, some results for 3D driven-cavity flow are given to demonstrate the ease with which the modified Shepard's interpolation can be applied to 3D flows.

### 4.1. Results of 2D Driven-Cavity Flow with Re = 5000

#### 4.1.1. *Problem Formulation and Solution Algorithm*

The Navier–Stokes equations in terms of curvilinear coordinates for 2D flow in a square cavity (see Fig. 3) can be written as

$$\begin{aligned}\omega_t & + \frac{y_\eta(u-x_t) - x_\eta(v-y_t)}{J}\,\omega_\xi \\ & + \frac{-y_\xi(u-x_t) + x_\xi(v-y_t)}{J}\,\omega_\eta \\ & = \frac{1}{\mathrm{Re}}\frac{1}{J^2}(\alpha\omega_{\xi\xi} - 2\beta\omega_{\xi\eta} \\ & \quad + \gamma\omega_{\eta\eta} + \sigma\omega_\eta + \tau\omega_\xi)\end{aligned} \tag{4.1}$$

$$\frac{1}{J^2}(\alpha\psi_{\xi\xi} - 2\beta\psi_{\xi\eta} + \gamma\psi_{\eta\eta} + \sigma\psi_\eta + \tau\psi_\xi) = -\omega, \tag{4.2}$$

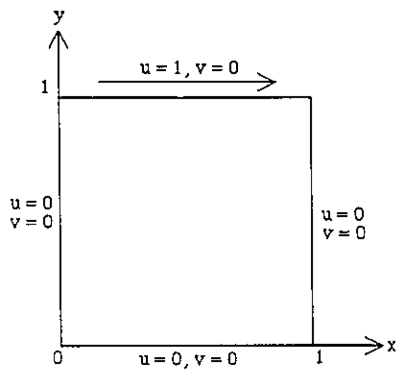

**FIG. 3.** 2D driven cavity.

where $u = \psi_y$, $v = -\psi_x$, and $\omega = v_x - u_y$. Here, $(x, y)$ and $(\xi, \eta)$ are coordinates of the physical and logical spaces, respectively. $J$ is the Jacobian of the transformation from physical space to logical space; $x_t$ and $y_t$ are the grid speed terms; $x_\xi$, $y_\xi$, $x_\eta$, $y_\eta$, $\alpha$, $\beta$, $\gamma$, $\sigma$, and $\tau$ are metric terms. The boundary conditions are given by

$$\begin{aligned} u = v &= 0, && \text{when} \quad x = 0 \text{ or } 1, \\ u = v &= 0, && \text{when} \quad y = 0, \\ u = 1, v &= 0, && \text{when} \quad y = 1. \end{aligned} \tag{4.3}$$

Equations (4.1) and (4.2) were discretized by applying a three-point central difference to spatial derivatives in $\omega$ and $\psi$ and a first-order backward difference to time derivatives in $\omega$. In order to ensure second-order accuracy of the converged solutions, vorticity boundary conditions at the wall were evaluated in a way similar to that of Ghia *et al.* [19]. The resulting nonlinear system of algebraic equations was solved by the modified strongly implicit procedure (MSI) which originated from Stone [21] and was subsequently extended to a nine-point finite-difference scheme by Schneider and Zedan [22]. Since steady-state solutions are of principal interest, adaption has to be done only occasionally during the solution procedure. Here the solution-adaptive method coupled with a continuation method is used.

The solution algorithm [17, 18] is outlined as follows:

1. Specify the initial value of Re. In order for solutions to converge faster, the calculation usually starts from a low Re.

2. Solve Eqs. (4.1)–(4.3) on a grid system using one-by-one iteration, i.e., one iteration for the stream-function equation followed by another iteration for the vorticity equation.

3. Calculate the weight functions for the adaptive-grid scheme [17, 18] based on the convergent (or reasonable intermediate) solutions of step 2.

4. Generate a new grid by solving the grid equations.

5. Map values of $\omega$, $\psi$, $u$, and $v$ from the old grid to the new grid.

6. Increase the value of Re.

7. Repeat steps 2 through 6 until the solution converges for the desired Re.

Because one-by-one iteration is used in this study, for 2D driven-cavity flow, the boundary values of vorticity at the walls have to be damped as:

$$(\omega_B)^{k+1} = \lambda(\omega_B)^{k+1} + (1 - \lambda)(\omega_B)^k \tag{4.4}$$

with $\lambda = 0.15$ [23]. Note also that the initial grid used can be either a uniform grid or a nonuniform grid based on previous knowledge of the flow. Also, if desired, the grid can be adapted further after the final solution is obtained; then Eqs. (4.1)–(4.3) are solved once more. Finally, singular value decomposition was used to solve the weighted least-square problem [24] of the modified Shepard's interpolation used in step 5.

### 4.1.2. *Convergence Criteria*

Since the principal interest here is the steady-state solution, the grid-speed terms in Eq. (4.1) are set to zero for the reason already mentioned in the Introduction. The solution procedure starts from Re = 400 on an $81 \times 81$ uniform grid and then Re is increased to 1000, 3200, and 5000. (Solution data for the intermediate solutions during the adaptions is available in the literature for comparison (Ghia *et al.* [19]).) The convergence criteria for both the Navier–Stokes solver and the grid-generation solver is defined as

$$\text{TOL} = \max_{i,j} \frac{|(\varphi_s)_{ij}^{k+1} - (\varphi_s)_{ij}^k|}{|(\varphi_s)_{ij}^{k+1}|_{\max}} < \varepsilon, \tag{4.5}$$

where $k$ is the iteration number; $\varphi_s$ can be $\omega$ and $\psi$ for the Navier–Stokes solver and $x$ or $y$ for the grid-generation solver. Here $\varepsilon$ is set to $5 \times 10^{-5}$ both for the grid-generation solver during the adaption stage and for the intermediate solutions of the Navier–Stokes solver. The value of $\varepsilon$ for the final converged solutions is $10^{-5}$ for both the grid-generation and Navier–Stokes solvers. In addition, root-mean-square residuals (RMSR), as suggested by Benjamin and Denny [23], are used to monitor errors during iteration in the Navier–Stokes solver. RMSR is set to $5 \times 10^{-3}$ and $10^{-4}$ for the intermediate and the final solutions, respectively. Note that these values are obtained after normalizing the Navier–Stokes equations.

### 4.1.3. *Numerical Accuracy*

In order to ensure that the numerical accuracy of the new interpolation method described in Section 2 is at least one

**TABLE I**

Errors Due to Two Mappings

| | $\mathrm{ISP}_q$ | $\mathrm{ISP}_w$ | $\omega$ | $\psi$ | $u$[a] | $v$[a] |
|---|---|---|---|---|---|---|
| Generalized bilinear interpolation | | | 6.7045E − 1 | 1.0624E − 4 | 2.3551E − 1 (1.0146E − 2) | 2.1000E − 1 (2.4876E − 3) |
| Proposed modified | 3 | 3 | 3.4077E − 2 | 1.1193E − 6 | 1.2346E − 2 (1.1880E − 4) | 7.2399E − 5 (1.0467E − 4) |
| Shepard's | 3 | 5 | 2.1219E − 1 | 9.6497E − 6 | 1.2353E − 2 (8.2450E − 4) | 7.8887E − 4 (1.1746E − 4) |
| interpolation | 5 | 3 | 1.1072E − 1 | 3.7471E − 6 | 1.2348E − 2 (2.8248E − 4) | 3.0027E − 4 (3.2859E − 4) |
| | 5 | 5 | 1.7010E − 1 | 5.2584E − 6 | 1.2348E − 2 (4.9486E − 4) | 3.6264E − 4 (6.5553E − 4) |

[a] Data in parentheses are obtained from the stream function.

order higher than the popular generalized bilinear interpolation, the following test was performed. The solutions based on the grid from the third adaption were used to generate a new grid and then mapped onto this new grid. (Similar results for other adaption stages are observed.) Since there are no analytic solutions available for comparison, the data on the new grid was then remapped onto the previous grid and the results compared with the original data from this grid. The root-mean-square norm was used to calculate the error; that is,

$$\left[\frac{\sum_{i,j}(\text{original data-data obtained after two mappings})^2}{\text{number of grid points}}\right]^{1/2}.$$

The results of Table I confirm that the modified Shepard's interpolation is in general (except for some values of the vorticity) at least one order of accuracy higher than the generalized bilinear interpolation. The results for $\mathrm{ISP}_q = \mathrm{ISP}_w = 3$ are the best. The possible explanation is that the stencil of the finite-difference method used in this study is a nine-point "star" (results from cross-derivative terms) which has the same structure as the proposed modified Shepard's interpolation with $\mathrm{ISP}_q = \mathrm{ISP}_w = 3$. Note that the error in $u$ is insensitive to the different values of $\mathrm{ISP}_q$ and $\mathrm{ISP}_w$ used. Here the largest contribution to the errors is from the two upper corners and this dominates the calculation of the root-mean-square norm.

The errors in stream function are small for both interpolation schemes. This is due to the fact that the stream-function profile is smooth and can be fit very well by either method. The results of Table I also show that the values of $u$ and $v$ calculated from the stream function are better than or comparable to those of the modified Shepard's interpolation. Thus, this method of calculation for $u$ and $v$ is adopted in this study.

Some results of driven-cavity flow for high Re flow have been shown; e.g., Benjamin and Denny [23], Ghia *et al.* [19], and Schreiber and Keller [20]. From among these, the results of Ghia *et al.* were selected in this study for comparison since detailed information is available. Figures 4–6 show that good agreement between the calculations of Ghia *et al.* and the present method is obtained. (Except for those used in the grid plot, all the data used in the figures have been mapped onto the uniform grid by the modified Shepard's interpolation.) Note that most of the results in the literature do not show the results of vorticity along the moving wall. Numerical experiments show that it is the most sensitive part of the calculation. The results of Re = 5000 on an 81 × 81 grid in the present calculations match those of Ghia *et al.* almost exactly. Furthermore Figs. 7a and b demonstrate the resemblance between the grid and the corresponding equi-vorticity lines not only for the main flow pattern, but also for the sub-structures of the two lower and upper left corners.

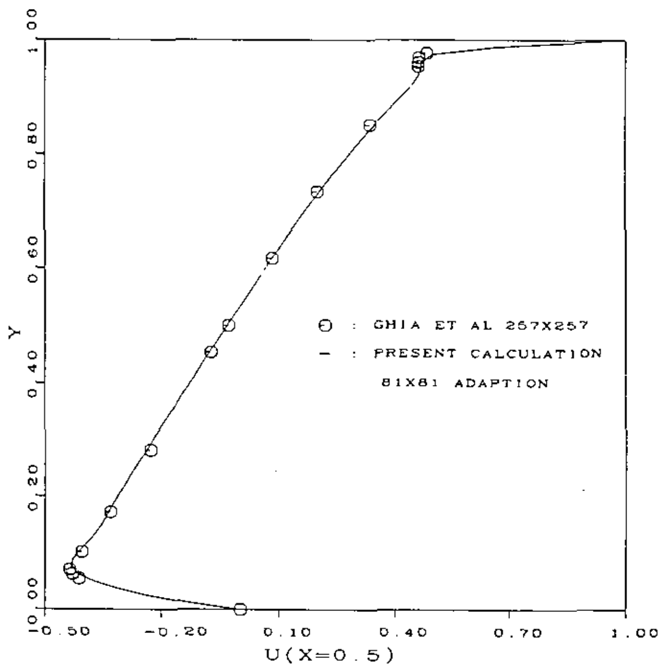

FIG. 4. $u$-velocity along the vertical line through the geometric center of cavity for the final converged solution of Re = 5000.

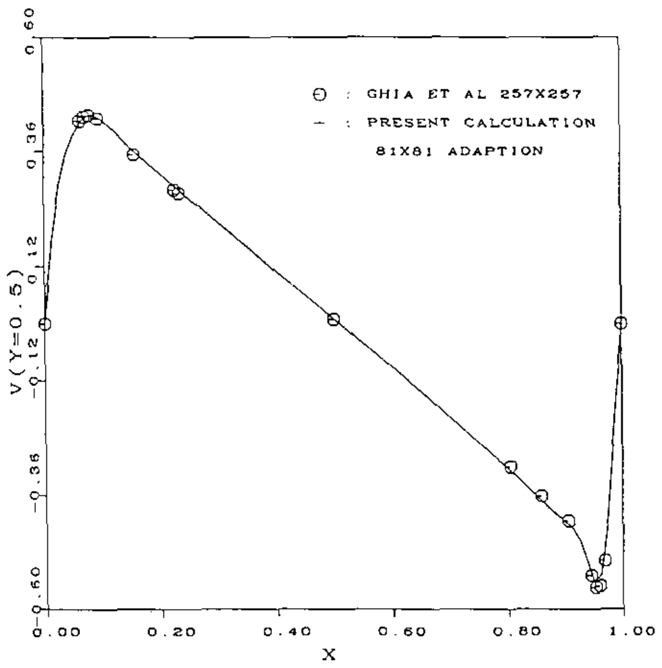

FIG. 5. $v$-velocity along the horizontal line through the geometric center of cavity for the final converged solution of Re = 5000.

### 4.1.4. *CPU Requirements*

The calculation was performed on a CRAY-XMP/18, using the CFT77 compiler on a single processor. The CPU time for Re = 5000 on an 81 × 81 grid (calculation started from a uniform grid for Re = 400 to the final converged

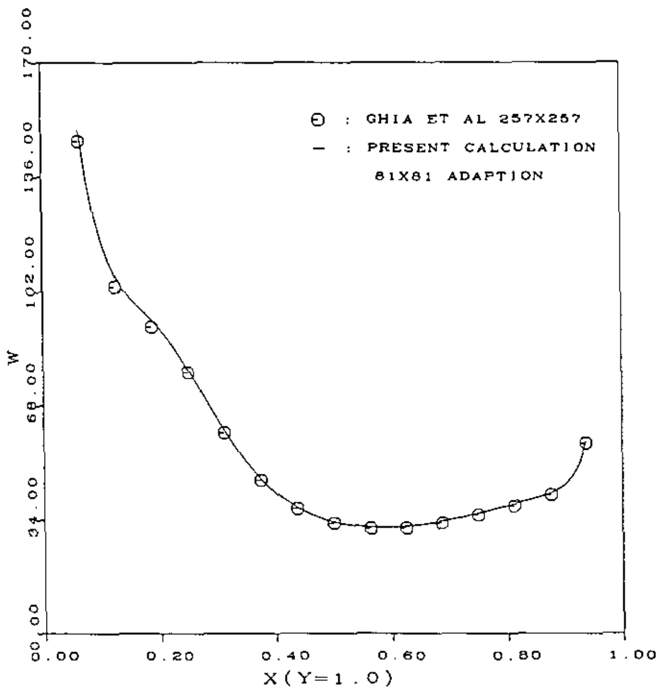

FIG. 6. Vorticity along the moving wall for the final converged solution of Re = 5000.

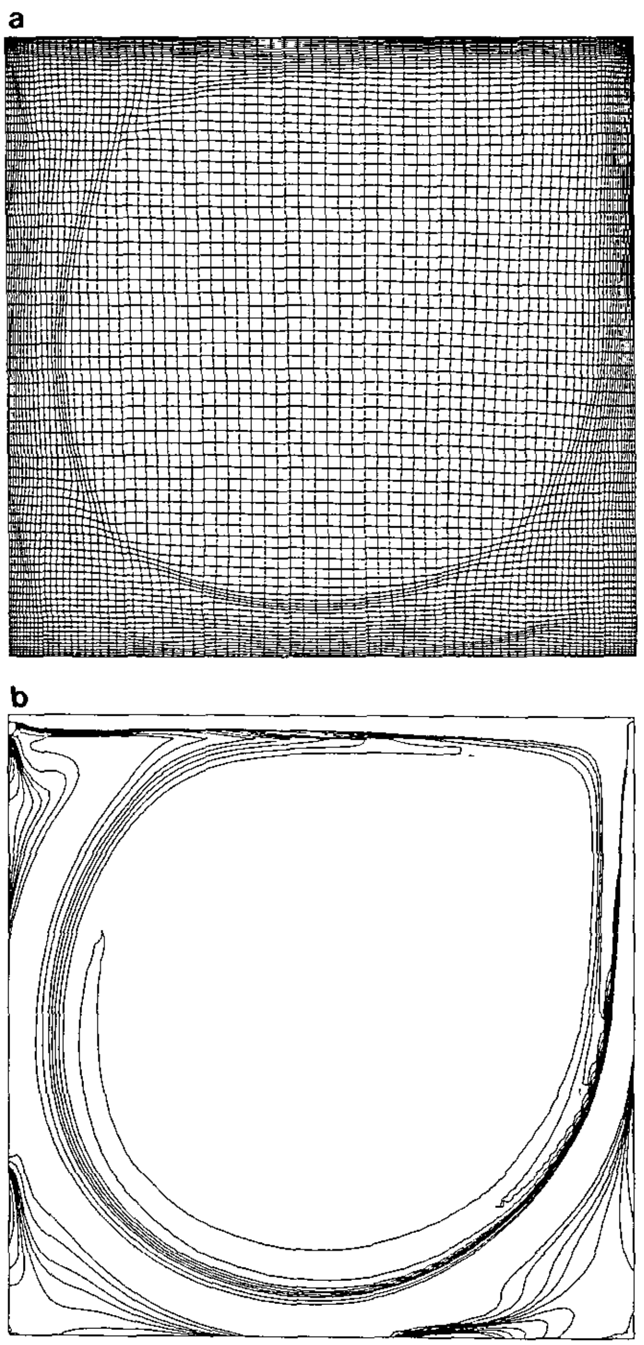

FIG. 7. (a) Final grid. (b) Equi-vorticity line for the final converged solution of Re = 5000.

solution for Re = 5000) is about 374 s. About 37% of the overall CPU time was spent on the adaptive-grid generation.

### 4.2. Results of 3D Driven-Cavity Flow

In order to demonstrate the ease with which the modified Shepard's interpolation and the solution algorithm similar to that presented in Section 4.1 can be applied to 3D flows,

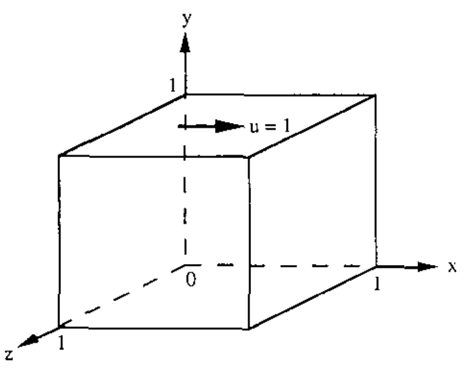

FIG. 8. 3D driven cavity.

some results for 3D driven-cavity flow are presented in this section (See Fig. 8).

Instead of using the stream-function/vorticity formulation, the vorticity/velocity formulation in 3D non-orthogonal coordinates [17, 18] is used. In other words, three velocity Poisson equations and three vorticity equations are solved by one-by-one iteration. As in 2D, a three-point central difference is applied to spatial derivatives for all equations and a first-order backward difference is applied to time derivatives in the vorticity equations. Again a false transient method [25] is used to improve the convergence of the solutions. The resulting nonlinear system of algebraic equations is solved by the MSI procedure which is designed for the seven-point stencil [26].

Although 3D driven-cavity flow has become an active subject in incompressible-flow calculation in the last decade [27–30], there are no results which are based on a very fine grid calculation available. In order to compare with the results obtained from the solution-adaptive method on a $21 \times 21 \times 21$ grid, a solution is also obtained for a $41 \times 41 \times 41$ stretched grid with different stretching parameters. The stretching function [31] is

$$x_i(\xi^i)=\frac{(\beta+1)((\beta+1)/(\beta-1))^{2(\xi^i-1)/(N^i-1)-1}-\beta+1}{2[1+((\beta+1)/(\beta-1))^{2(\xi^i-1)/(N^i-1)-1}]} \tag{4.6}$$

with $i=1$, 2, 3; $x_i$ are the physical coordinates and $N^i$ are the grid numbers along the $\xi^i$-coordinates. $\beta$ is the stretching parameter whose value has been chosen as $\infty$, 1.195, and 1.118 in this study. Note that $\beta=\infty$ corresponds to a uniform grid and the smallest step sizes near the wall are $1.1403 \times 10^{-2}$ and $0.8610 \times 10^{-2}$ for $\beta=1.195$ and 1.118, respectively. Both Re = 100 and 400 are performed but only results of Re = 400 are presented here.

Results of Re = 100, obtained from the solution-adaptive method on a $21 \times 21 \times 21$ grid, agree very well with those of the stretched grids on a $41 \times 41 \times 41$ grid basis. This is not surprising; since the flow field of Re = 100 is rather smooth,

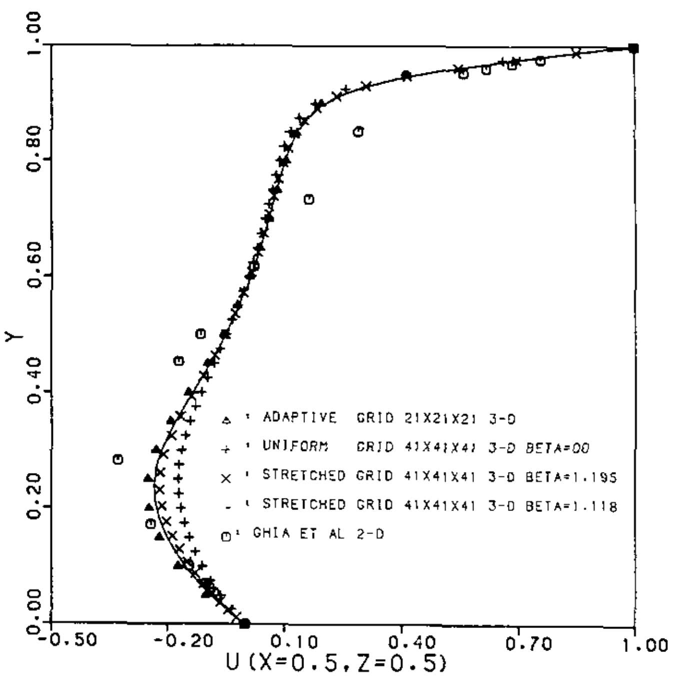

FIG. 9. The variation of $u$ component of velocity at Re = 400, $y = 0.5$, and $z = 0.5$ as a function of $x$.

even the calculation on a $21 \times 21 \times 21$ grid can produce reasonable solutions. But this is not the case for Re = 400. Figures 9–10 show the variations of the $u$ and $v$ velocity components in the center plane of the cubic cavity. For a $41 \times 41 \times 41$ grid, the difference between the solutions obtained from the uniform grid ($\beta = \infty$) and those from

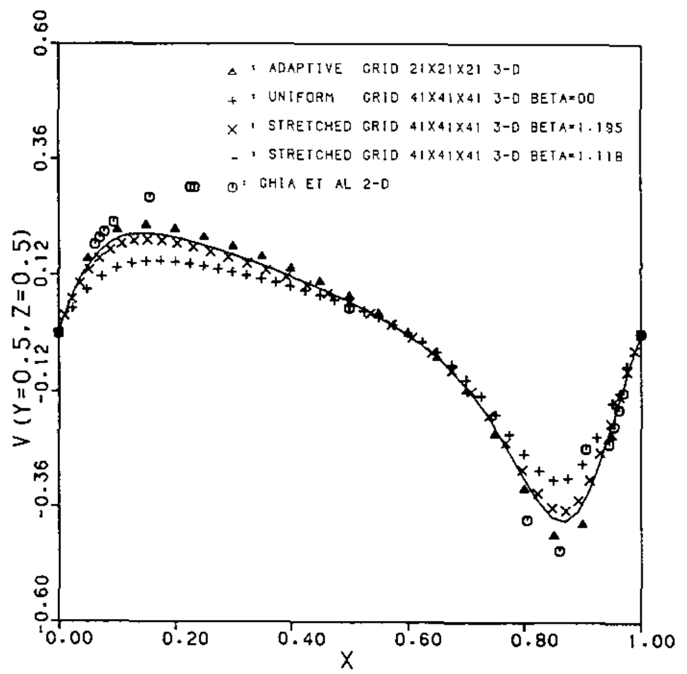

FIG. 10. The variation of $v$ component of velocity at Re = 400, $x = 0.5$, and $z = 0.5$ as a function of $y$.

stretched grids ($\beta = 1.195$ and 1.118) are quite large. This is due to the fact that the step size near the wall for the uniform grid is still too coarse for Re = 400. But, as long as the grid is refined near the wall, the solutions start to collapse together as can be seen from the $\beta = 1.195$ and 1.118 calculation. Thus, the solutions obtained on a $41 \times 41 \times 41$ grid with $\beta = 1.118$ are used to compare with those from the solution-adaptive method on a $21 \times 21 \times 21$ grid. Figures 9–10 show that good agreement is obtained. Note that the results of the solution-adaptive method have been mapped onto a uniform grid using the modified Shepard's interpolation.

Due to the computational cost, all the parameters (e.g., the relaxation parameter used in the MSI method) used in this study are not optimal. Thus, it is difficult to give a fair comparison in order to comment on the efficiency of the method for the cases presented in this section. However, some idea of the relative CPU time spent on the calculation with and without the adaptive grid (with the same quality of solutions) can be obtained from the numerical experiments completed thus far. Roughly speaking, the CPU time needed for the solution-adaptive method on a $21 \times 21 \times 21$ grid is at least 10 times smaller than that required for the stretched grids for Re = 400.

## 5. CONCLUSIONS

A new implementation of a modified Shepard's interpolation has been presented for the structured grids of CFD; particularly its use in the remapping step of a solution-adaptive method has been demonstrated for both 2D and 3D incompressible driven-cavity flows. With this new extension, the method can now be applied not only to scattered data, but also to structured adaptive-grid methods, composite-grid methods, Euler–Lagrangian methods, moving-boundary problems, multi-level multigrid problems, etc. Also, the simplified version provides a simpler way to determine which neighboring points are to be included in the local least-square quadratic fit when used for locally rectangular grids; the search method needed in the original modified Shepard's interpolation for scattered data is potentially very time consuming. Results show that the proposed modified Shepard's interpolation has high accuracy and is easy to implement for both 2D and 3D. The most important advantage of the new method, however, is robustness. The popular trilinear interpolation can fail during a search in a 3D calculation as pointed out in Section 1.1; in 2D, this is not a problem. For 3D adaptive methods using remapping, this new method will not fail.

## ACKNOWLEDGMENTS

We thank Dr. Gregory M. Nielson, Department of Computer Science and Engineering, Arizona State University, for helpful discussion. This work was supported by the Office of Naval Research under Contract N00014-86-K-0066 and in part by the NSF and AFOSR Contract DMS-9116930.

## REFERENCES

1. J. Saltzman and J. Brackbill, in *Numerical Grid Generation*, edited by J. F. Thompson (North–Holland, Amsterdam, 1982), p. 865.
2. K. Nakahashi and G. S. Deiwert, *AIAA J.* **24**, No. 6, 948 (1986).
3. D. A. Anderson, in *Advances in Grid Generation*, edited by K. N. Ghia and U. Ghia (ASME-FED, ASME, New York, 1983), p. 1.
4. J. F. Thompson, *Appl. Numer. Math.* **1**, 3 (1985).
5. P. R. Eiseman, *Comput. Methods Appl. Mech. Eng.* **64**, 321 (1987).
6. C. W. Mastin, in *Numerical Grid Generation in Computational Fluid Mechanics'88*, edited by S. Sengupta, J. Hauser, P. R. Eiseman, and J. F. Thompson (Pineridge, Swansea, UK, 1988), p. 63.
7. D. Seldner and T. Westermann, *J. Comput. Phys.* **79**, 1 (1988).
8. D. Shepard, in *Proceedings, 23rd National Conference of ACM, 1968*, p. 517.
9. W. J. Gordon and J. A. Wixom, *Math. Comput.* **32**, No. 141, 253 (1978).
10. R. E. Barnhill, R. P. Dube, and F. F. Little, *Rocky M. J. Math.* **13**, No. 2, 365 (1983).
11. L. L. Schumaker, in *Approximation Theory II*, edited by G. G. Lorentz, C. K. Chui, and L. L. Schumaker (Academic Press, New York, 1976), p. 203.
12. R. Franke and G. Nielson, *Int. J. Numer. Methods Eng.* **15**, 1691 (1980).
13. R. J. Renka, *ACM Trans. Math. Software* **14** (2), 139 (1988).
14. R. E. Barnhill, in *Mathematical Software III*, edited by J. R. Rice (Academic Press, New York, 1977).
15. J. F. Thompson, in *Numerical Grid Generation in Computational Fluid Mechanics '88*, edited by S. Sengupta, J. Hauser, P. R. Eiseman, and J. F. Thompson (Pineridge, Swansea, U.K., 1988), p. 87.
16. H. J. Kim and J. F. Thompson, AIAA-88-0311.
17. C. Y. Shen and H. L. Reed, *Comput. Fluids*, submitted.
18. C. Y. Shen, Ph. D. dissertation, Arizona State University, 1991.
19. U. Ghia, K. N. Ghia, and C. T. Shin, *J. Comput. Phys.* **48**, 387 (1982).
20. R. Schreiber and H. B. Keller, *J. Comput. Phys.* **49**, 310 (1983).
21. H. L. Stone, *SIAM J. Numer. Anal.* **5** (3), 530 (1968).
22. G. E. Schneider and M. Zedan, *Numer. Heat Transfer* **4**, 1 (1981).
23. A. S. Benjamin and V. E. Denny, *J. Comput. Phys.* **33**, 340 (1979).
24. W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical Recipes: The Art of Scientific Computing* (Cambridge Univ. Press, Cambridge, UK, 1968), p. 509.
25. K. H. Chen, private communication.
26. M. Zedan and G. E. Schneider, *AIAA J.* **21** (2), 295 (1983).
27. S. C. R. Dennis, D. B. Ingham, and R. N. Cook, *J. Comput. Phys.* **33**, 325 (1979).
28. R. K. Agarwal, in *Computers in Flow Predictions and Fluid Dynamics Experiments, Winter Annual Meeting of ASME, 1981*, edited by K. N. Ghia *et al.*, p. 73.
29. H. C. Ku, R. S. Hirsh, and T. D. Taylor, *J. Comput. Phys.* **70**, 439 (1987).
30. G. A. Osswald, K. N. Ghia, and U. Ghia, AIAA-87-1139.
31. D. A. Anderson, J. C. Tannehill, and R. H. Pletcher, *Computational Fluid Mechanics and Heat Transfer* (Hemisphere, McGraw–Hill, Washington, DC/New York, 1984), p. 247.