Journal of Computational Physics **179,** 469–494 (2002)
doi:10.1006/jcph.2002.7067

A Cartesian Grid Method with Transient Anisotropic Adaptation

F. E. Ham, F. S. Lien, and A. B. Strong

Department of Mechanical Engineering, University of Waterloo, Waterloo, Ontario, Canada N2L 3G1
E-mail: feham@sunwise.uwaterloo.ca, fslien@sunwise.uwaterloo.ca, astrong@sunwise.uwaterloo.ca

Received February 24, 2001; revised February 12, 2002

A Cartesian grid method with solution-adaptive anisotropic refinement and coarsening is developed for simulating time-dependent incompressible flows. The Cartesian grid cells and faces are managed using an unstructured data approach, and algorithms are described for the time-accurate transient anisotropic refinement and coarsening of the cells. The governing equations are discretized using a collocated, cell-centered arrangement of velocity and pressure, and advanced in time using the fractional step method. Significant savings in the memory requirement of the method can be realized by advancing the velocity field using a novel approximate factorization technique, although an iterative technique is also presented. The pressure Poisson equation is solved using additive correction multigrid, and an efficient coarse grid selection algorithm is presented. Finally, the Cartesian cell geometry allows the development of relatively simple analytic expressions for the optimal cell dimensions based on limiting the velocity interpolation error. The overall method is validated by solving several benchmark flows, including the 2D and 3D lid-driven cavity flows, and the 2D flow around a circular cylinder. In this latter case, an immersed boundary method is used to handle the embedded cylinder boundary. © 2002 Elsevier Science (USA)

Key Words: Cartesian grid methods; grid adaptation; local grid refinement.

1. INTRODUCTION

Cartesian grid methods with local refinement have been in the literature for a number of years. Their main advantages are in the simplification of the grid generation process and the ease with which local refinement of the grid can be accomplished through recursive cell-splitting operations. These methods have enjoyed the majority of their success when applied to the Euler equations, where they are commonly combined with an interpolation or cut-cell technique to handle embedded boundaries and compute the inviscid flow around complex geometrical configurations [1–3].

The extension of Cartesian grid methods with local refinement to the solution of viscous flow problems has been limited mainly by the resolution requirements imposed by viscous boundary layers, where highly anisotropic meshes aligned with the flow direction are most appropriate. Some authors have proposed using a hybrid of highly anisotropic body-fitted grids in the regions close to embedded bodies and Cartesian grids elsewhere [4], although this approach gives up much of the simplicity and elegance of the Cartesian grid method. A partial solution to the viscous boundary layer problem is the introduction of anisotropic refinement of the Cartesian cells. Courier has already experimented with this technique in the context of viscous flows, although found that the resulting meshes were sometimes too irregular to compute an accurate solution [5]. As Berger and Aftosmis correctly point out [6], the asymptotic limit of anisotropic refinement of Cartesian cells is not sufficient to produce boundary-layer zoning. When the direction of anisotropy is not aligned with one of the principal coordinate directions, the boundary layer resolution requirements may lead to a nearly isotropic refinement, reducing or completely canceling any savings related to the anisotropic refinement capabilities. In general, however, most boundaries and their associated boundary layers will be at least partially aligned with one of the principal coordinate directions, and even slightly anisotropic Cartesian cells have the potential to significantly reduce the number of unknowns, particularly in three dimensions.

The development of a Cartesian grid method with local anisotropic refinement (and coarsening) suitable for time-dependent viscous flow computations is the subject of the present contribution. The remainder of the paper is organized as follows. In Section 2, the Cartesian grid data structure is described. The present method adopts an unstructured data approach to manage the Cartesian cells and faces. Section 2 also presents algorithms for the anisotropic refinement and coarsening of the Cartesian cells. In Section 3 the numerical method is presented. The method uses a second-order finite volume discretization and a collocated cell-centered arrangement of variables. The fractional step technique [7, 8] is used to decouple the time advancement of the velocity field from the pressure. We show how the approximate factorization technique common to structured grid methods can be applied to the present Cartesian grids to advance the velocity field directly and yield substantial memory savings. The pressure equation is solved using additive correction multigrid, and a simple multigrid coarsening algorithm is presented. In Section 4, a solution-based grid adaptation criterion is developed. The regular geometry of the Cartesian cell allows the derivation of relatively simple analytic expressions for the optimal cell dimensions based on limiting the velocity interpolation error. In Section 5 the overall method is validated by solving several benchmark flows, including the 2D and 3D lid-driven cavity flows, and the 2D flow around a circular cylinder. In this latter case, an immersed boundary method is used to handle the embedded cylinder boundary. The paper is summarized in Section 6.

2. THE ADAPTIVE CARTESIAN GRID

Lohner [9] proposes three main ingredients for any adaptive refinement scheme:

1. an error indicator,
2. an optimal mesh criterion, and
3. an algorithm to refine and coarsen the mesh.

In the present section we address only the third item and assume for now that the cells requiring refinement or coarsening are known. A discussion of the first two items of this list, which actually identify the cells requiring refinement or coarsening, is presented in Section 4.

For the purposes of geometric clarity, the Cartesian cell data structure and various algorithms presented in this section are developed for two dimensions. In all cases, the extension to three dimensions is straightforward.

2.1. Data Structure

2.1.1. *Rationale*

The set of locally refined Cartesian cells and their associated data and neighbour connections are commonly managed in one of two ways: either a hierarchical parent–child tree structure or a completely unstructured approach. Coirier and Powell use the hierarchical tree structure in their Euler and Navier–Stokes solvers [1, 5], demonstrating that the tree structure provides a logical means of finding cell-to-cell connectivity and allows straightforward isotropic refinement and coarsening through tree growth and pruning. The research behind NASA's Cart3D inviscid flow project is also based on this hierarchical approach [3, 6, 10]. A fully unstructured data approach is more common when local refinement is used in the context of body-fitted structured grids. For example, see [11] for a general discussion or Seidl *et al.* [12] for a specific example.

The main argument against the fully unstructured approach is that of increased memory requirement—the fully unstructured approach requires that each cell store an array of references/pointers to its immediate neighbours. The cost of storing these neighbour references may be three or more times the memory overhead associated with a tree structure. Because both methods still have to store unknowns, geometrical information, and perhaps coefficients, the difference in terms of the total memory requirement may be much less substantial. Further, even though the tree traverses required to determine neighbour connectivity are based on logical recursive routines, the calculation time required for computing quantities involving the neighbours (e.g., residual calculations) will be greater than for the fully unstructured approach, where the neighbour references are explicitly stored.

Some authors have suggested [5] or demonstrated [10] that the grid hierarchical tree structure is amenable to multigrid. While the tree structure may be useful for developing an efficient grid coarsening strategy, the actual memory overhead associated with the multigrid implementation will be about the same in both cases.

Finally, we point out that the solution of viscous incompressible flows requires that careful attention be paid to the velocity–pressure coupling to prevent nonphysical oscillations in the solution. This is normally accomplished by using variants of either the staggered grid method of Harlow and Welch [13] or the collocated method of Rhie and Chow [14]. In either case, the calculation and storage of face-based data are required. The storage and management of faces between cell neighbours would seem better suited to the fully unstructured data approach. Based on this reasoning, the present method uses a fully unstructured approach to manage the cell and face data.

2.1.2. *Description*

Consider the rectangular domain shown in Fig. 1a with dimensions L_x by L_y, and southwest corner at (x_0, y_0). The domain is initially covered by a single Cartesian cell, which we identify using the standard structured grid indexing as cell $(i, j) = (0, 0)$. The cell is

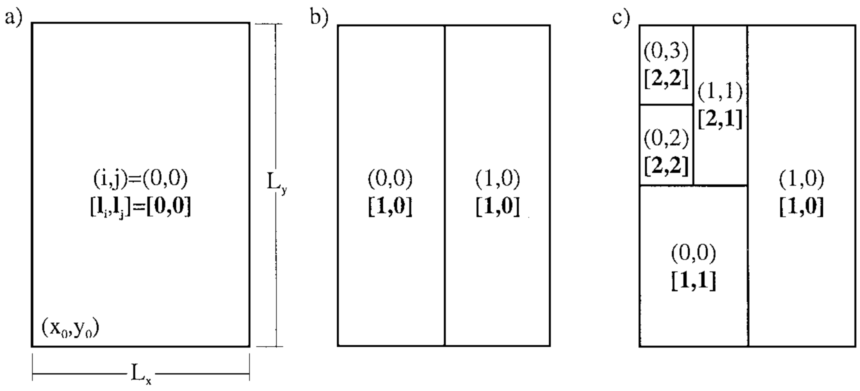

FIG. 1. Cartesian cell indices (i, j) and $[l_i, l_j]$ for (a) the initial mesh of one cell, (b) the mesh after one anisotropic refinement in the x-direction, and (c) the mesh after several anisotropic refinements.

also identified by a second set of indices that define its level of refinement in each of the coordinate directions—in this case $[l_i, l_j] = [0, 0]$, where level zero represents the coarsest possible cell. Square brackets are used to distinguish between the two indices. Because the cells are managed using a fully unstructured approach, each cell must explicitly store its own (i, j) and $[l_i, l_j]$ indices.

Most Cartesian cell methods starts from this initial discretization of the domain and then perform a few isotropic refinements to produce a suitably refined initial mesh. The present method proceeds in a similar manner; however, only anisotropic refinements are allowed. This approach does not preclude isotropic refinement, which can of course be achieved by two successive anisotropic refinements in two different directions. Figure 1b shows the domain after a single anisotropic refinement in the x-direction. In this case, the refinement level in the x-direction has been increased in both cells to $l_i = 1$. The (i, j) indices of each cell store the standard structured grid indices as if the entire grid were at the refinement level of the cell. Figure 1c shows the mesh and associated indices after a few more anisotropic refinements have been performed.

Using the stored index information, the centroid (x_c, y_c) and dimensions $(\Delta x, \Delta y)$ of each cell can be calculated as follows:

$$(x_c, y_c) = \left(x_0 + (i + 1/2)\frac{L_x}{2^{l_i}}, y_0 + (j + 1/2)\frac{L_y}{2^{l_j}}\right) \tag{1}$$

$$(\Delta x, \Delta y) = \left(\frac{L_x}{2^{l_i}}, \frac{L_y}{2^{l_j}}\right). \tag{2}$$

Because the integer powers of 2 in the above equations can be computed very efficiently, each cell's geometric data can be calculated on an as-needed basis, resulting in memory savings and guaranteeing consistency between geometric data and index information.

2.2. Mesh Refinement and Coarsening

By treating the cells in a fully unstructured manner and making anisotropic refinement and coarsening the norm, the complexities of introducing cell anisotropy into an isotropic hierarchical tree data structure are avoided. Unstructured treatment of the cells also allows for the anisotropic coarsening of the mesh in a different direction than the inverse of the most recent refinement.

In the refinement and coarsening routines of the present work, the only restriction imposed is that no cell can have more than two neighbours in any one direction (or four neighbours in three dimensions). This is equivalent to saying that the absolute difference in y-refinement level, l_j, between two cells that are east/west neighbours (i.e., share a face for which the normal has only an x-component) must be less than or equal to 1. For north/south neighbours, the absolute difference in x-refinement level, l_i, must be less than or equal to 1. No limit is imposed on the difference in x-refinement level between east/west neighbours or the difference in y-refinement level between north/south neighbours. Although these additional restrictions would result in a smoother mesh, this was not found to be necessary.

The procedure for refining a cell in the x-direction is given below as **refineX**. The procedure returns the boolean value TRUE upon successful refinement of the cell; otherwise it returns FALSE.

```
boolean refineX (Cell $c$)
1.  Make sure we are OK to x-refine
    if ($c$ is set to be $x$-coarsened) OR ($c$ has already been $x$-refined)
        return FALSE
    end if
2.  Try to refine any neighbours that are preventing our own refinement
    for each north/south Cell neighbour $c_{nb}$
        if ($l_{i,nb} == l_i - 1$) AND (refineX($c_{nb}$) == FALSE)
            return FALSE
        end if
    end for
3.  c becomes the refined cell to the west. Its indices are modified as follows:
    $l_i = l_i + 1, i = 2i$
4.  Add a new cell, $c_{new}$, to the east. Its indices are set as follows:
    $l_{i,new} = l_i, i_{new} = i + 1$
    $l_{j,new} = l_j, j_{new} = j$
5.  Modify neighbour connectivity, add/modify faces, interpolate cell and
    face data
6.  Mark Cells c and $c_{new}$ as x-refined, and return
    return TRUE
end refineX
```

A similar refinement routine is required for the y-direction. The details of modifying the neighbour connectivity and adding new faces are tedious but not complex. The conservative interpolation of the cell and face data will be addressed in the following section on the numerical method.

The anisotropic coarsening procedure in the x-direction is now given as **coarsenX**. Coarsening requires slightly more checking to identify a neighbour that can be combined with the cell in question to produce a valid coarse cell. This checking is performed as part of step 2 using the integer division ($i/2 == i_{nb}/2$). Integer division rounds down when there is a remainder. For example, the two cells shown in Fig. 1b have $i = 0$ and $i = 1$, respectively. Using integer division, both cells have $i/2 = 0$ and thus form a valid pair for coarsening.

```
boolean **coarsenX** (Cell $c$)
1.  *Make sure we are OK to x-coarsen*
    if ($c$ is set to be $x$-refined) OR ($c$ has already been $x$-coarsened)
        return FALSE
    end if
2.  *Cycle through our neighbours to find a cell we can x-coarsen with*
    for each east/west Cell neighbour $c_{nb}$
        if ($i/2 == i_{nb}/2$) AND ($c_{nb}$ can be coarsened) AND
            ($l_i == l_{i,nb}$) AND ($l_j == l_{j,nb}$)
3.          *Make sure none of the north/south neighbours of either c or*
            *$c_{nb}$ prevent the coarsening*
            if all north/south neighbours, $c_{n/s}$, have ($l_{i,n/s} <= l_i$)
4.              *Cell c becomes the coarse cell, $c_{nb}$ is deleted*
                $l_i = l_i - 1, i = i/2$
5.              *Modify neighbour connectivity, remove/modify faces,*
                *interpolate cell and face data*
6.              *Mark Cell c as x-coarsened, and return*
                return TRUE
            end if
        end if
    end for
return FALSE
end **coarsenX**
```

Note that, unlike the refinement routine, the coarsening routine is not called recursively to try and modify north/south neighbours that may be preventing the coarsening from going ahead (because of the maximum neighbour rule). This difference was found to reduce the tendency of certain parts of the mesh to oscillate between coarsening and refinement, with a bias toward refinement.

A similar coarsening routine is required for the y-direction. As with refinement, interpolation of the cell and face data will be addressed in the following section on the numerical method.

3. NUMERICAL METHOD

The numerical method of the present contribution uses a collocated treatment of variables, with all unknowns stored at the Cartesian cell centroids. The governing Navier–Stokes equations for unsteady incompressible flow are discretized using the finite-volume method with second-order, linearly exact discretizations for all fluxes. Time advancement uses the fractional step method, which decouples the solution of the velocity field from the pressure. In each time step, the velocity field is first advanced, followed by the solution of a pressure Poisson equation to enforce continuity. The overall algorithm is based on the method of Kim and Choi for unstructured grids [8], although we make modifications to the flux discretizations they propose that make use of the simplifications engendered by the Cartesian cell mesh and allow the use of approximate factorization to advance the velocity field.

3.1. Viscous Flux Discretization

As pointed out by Coirier and Powell [5], a major limitation of Cartesian cell methods is that the viscous flux discretization can have detrimental effects on both the accuracy and convergence in regions where the mesh is not smooth. Inaccuracy occurs when the discretization is not linearity preserving, and convergence problems are related to nonpositivity of the reconstruction. To minimize these problems, Coirier and Powell used a "diamond path scheme"—basically a divergence-theorem-based reconstruction where the integration path in two dimensions forms a diamond about the face in question. Even with this scheme, nonpositivity can still lead to convergence problems. Other authors have recently proposed modifications to the diamond path to reduce these problems [4].

In our opinion, the linearly exact interpolations required by the diamond path scheme render it computationally costly and difficult to implement, particularly in three dimensions. In addition, the scheme does not make use of the underlying Cartesian geometry to simplify the reconstruction. For these reasons, the present work considers two other viscous flux discretizations. The first is the fully unstructured viscous flux discretization proposed by Zwart *et al.* [15]. The second is developed from the "auxiliary node" concept of Ferziger and Peric [11] for unstructured grids. Both discretizations are second-order accurate and linearly exact. Both schemes also require the velocity gradients at the cell centroids. In the present work, these gradients are calculated using a least-squares reconstruction.

The details of the two diffusive flux discretizations are now presented. Consider the east/west cell neighbours and their shared face shown in Fig. 2. For incompressible flow with viscosity ν, the viscous flux of x-momentum through face f with unit normal $\hat{n}$ and area A_f can be expressed

$$F_f^d(u) = -\nu(\nabla u \cdot \hat{n})A_f. \tag{3}$$

Using the second-order, linearly exact discretization of Zwart *et al.*, this flux can be approximated at the face as

$$F_f^{d,1}(u) = -\nu\left[(\hat{n}\cdot\hat{s})\frac{u_Q - u_P}{\Delta s} + \overline{\nabla u}\cdot(\hat{n} - (\hat{n}\cdot\hat{s})\hat{s})\right]A_f, \tag{4}$$

where $\hat{s}$ is the unit vector along the line joining the two cell centroids, Δs is the distance between the centroids, and $\overline{\nabla u}$ is the average of the gradients at the cell centroids. For the Cartesian cell geometry shown in Fig. 2, the face normal has components $(n_x, n_y) = (1, 0)$, and Eq. (4) simplifies to

$$F_f^{d,1}(u) = -\nu\frac{s_x}{\Delta s}(u_Q - u_P)A_f - \nu\left(\overline{\frac{\partial u}{\partial x}}s_y^2 - \overline{\frac{\partial u}{\partial y}}s_x s_y\right)A_f. \tag{5}$$

In Eq. (5), the flux expression has been split into two parts, the first involving the

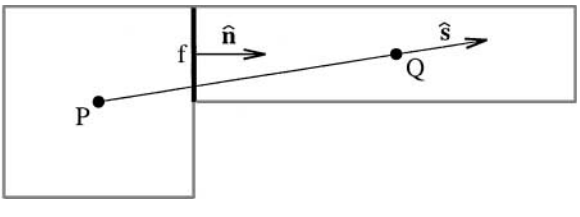

FIG. 2. Two-dimensional cell and face schematic for the viscous flux discretization of Zwart *et al.* 15].

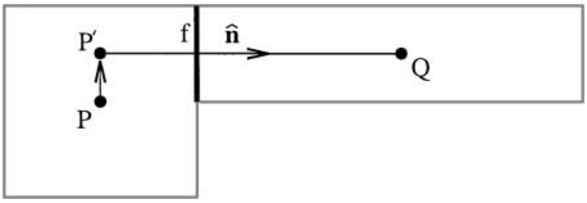

FIG. 3. Cell and face schematic for an alternative viscous flux discretization based on the auxiliary node concept.

cell-centered unknowns, and the second involving the cell-centered gradients. We will refer to the first part as the active part, and the second part as the lagged part,

$$F_f^d(u) = F_f^{d,Active}(u) + F_f^{d,Lagged}(u). \tag{6}$$

On Cartesian grids with local grid refinement, it is common for the $\hat{s}$ and $\hat{n}$ vectors to be far from aligned. Inspection of Eq. (5) shows that this will significantly reduce the magnitude of the active part of the flux expression.

An alternative flux expression can be developed using the auxiliary node concept of Ferziger and Peric [11]. Figure 3 shows the same pair of cells considered previously with the auxiliary node P' introduced at the point (x_P, y_f). A linearly exact, second-order accurate approximation for the viscous flux of x-momentum through face f for this particular configuration is then

$$F_f^{d,2}(u) = -\nu\left(\frac{u_Q - u_{P'}}{x_Q - x_P}\right)A_f. \tag{7}$$

Using the cell-center gradients to approximate the values at the auxiliary nodes, the flux expression can be written more generally for an arbitrary east/west face as

$$F_f^{d,2}(u) = -\nu\left(\frac{u_Q - u_P}{x_Q - x_P}\right)A_f - \nu\frac{\frac{\partial u}{\partial y}\big|_Q(y_f - y_Q) - \frac{\partial u}{\partial y}\big|_P(y_f - y_P)}{(x_Q - x_P)}A_f. \tag{8}$$

Once again, the flux expression has been split into an active and a lagged part. Note also that, in all cases, at least one of the gradient terms in the lagged part will be zero (because y_f and either y_P or y_Q will be the same).

The two discretization schemes described by Eqs. (5) and (8) are identical when the vectors $\hat{s}$ and $\hat{n}$ are aligned, corresponding to the case of Cartesian cells with no relative refinement. When these vectors are not aligned, however, the magnitude of the active part of the scheme of Zwart *et al.* is always less than that of the auxiliary node method. This can be shown by taking the ratio of the active parts of the two schemes,

$$\left.\frac{F_f^{d,1}}{F_f^{d,2}}\right|_{active} = \frac{1}{1 + (s_y/s_x)^2} = \frac{1}{1 + \tan^2\theta}, \tag{9}$$

where θ is the angle between $\hat{s}$ and $\hat{n}$.

Based on the numerical experiments reported in the next sections, we came to the following conclusions on the relative suitability of the two schemes for Cartesian meshes.

Both schemes gave similar results when the underlying time advancement was semiimplicit and approximate factorization was used to directly advance the velocity field (see later section). When a fully implicit approach was used and the velocity field was advanced using an iterative technique, converged solutions on adapted Cartesian meshes could only be achieved using the second auxiliary node flux discretization. The larger implicit part of the auxiliary-node method seems critical for numerical stability on highly adapted Cartesian meshes.

3.2. Convective Flux Discretization

While not specifically considered in this paper, one potential application of the present method is to the direct numerical simulation and large eddy simulation of turbulent flows. It is well known that even higher order upwind discretizations of the convective terms can introduce numerical dissipation into the flow solution, particularly at the smallest resolved scales. While the stabilizing effect of this dissipation can be beneficial for the smoothly varying velocity fields of laminar or Reynolds-averaged Navier–Stokes simulations, it can significantly degrade the accuracy of unsteady turbulent flow simulations.

With this future application in mind, the present method uses a symmetric ("central-difference") discretization for the convective flux based on the auxiliary node technique discussed in the previous section. First, define a linearly exact interpolation operator, which can be applied to cell-centered data to get a face value. Considering the pair of east/west cell neighbours and their shared face shown in Fig. 3, the interpolation operator is defined as

$$\bar{\phi}_f = \frac{((x_f - x_P)\phi_Q + (x_Q - x_f)\phi_P)}{(x_Q - x_P)} + \frac{(x_f - x_P)(y_f - y_Q)\frac{\partial\phi}{\partial y}\big|_Q + (x_Q - x_f)(y_f - y_P)\frac{\partial\phi}{\partial y}\big|_P}{(x_Q - x_P)}, \tag{10}$$

where ϕ is any cell-centred quantity. Note that this interpolation operator has also been divided into active and lagged parts. A similar interpolation operator can be defined for north/south faces. Using this operator, a second-order linearly exact approximation for the convective flux of x-momentum through face f is then

$$F_f^c(u, U) = U_f A_f \bar{u}_f, \tag{11}$$

where U_f is the mass-conserving face-normal velocity, calculated using the Rhie and Chow-type interpolation described in the following section.

3.3. Overall Numerical Method

With the flux discretizations described, we now summarize the overall numerical method. Two separate approaches are considered. In addition to the fully implicit linearized discretization of Kim and Choi, we consider a semiimplicit approach similar to that of Zang *et al.* [16], where the active part of the viscous terms is treated implicitly, and the convective terms and lagged part of the viscous terms are treated explicitly using two-level Adams–Bashforth. Although this semiimplicit approach has the disadvantage of introducing a numerical stability limit on the computational time step, there are several advantages that may offset this. (1) The explicit treatment of the lagged terms means that the gradients at the cell centroids, which come into these terms, are calculated once at the start of each time step, rather than throughout the iterative process. (2) The semiimplicit approach allows the use of the approximate factorization technique to advance the velocity field. Approximate factorization is computationally efficient and saves significantly on memory because it does not require the storage of neighbour coefficients for the momentum equations in each cell. Its application to Cartesian meshes with local refinement is presented in a later section.

3.3.1. *Fully Implicit Approach*

Although the fully implicit approach of the present work closely follows the method described by Kim and Choi [8], it is repeated here for completeness.

After application of the fractional step method, the divergence theorem, and then discretizing the resulting fluxes using the schemes defined previously, the following equations must be solved to calculate the velocity and pressure at the next time level, u_i^{n+1} and p^{n+1},

$$\frac{\delta\hat{u}_i}{\Delta t}\mathrm{Vol} + \frac{1}{2}\sum_f\left(F_f^c(\delta\hat{u}_i, U^n) + F_f^c\left(u_i^n, \overline{\delta\hat{u}_j n_j}\right)\right) + \frac{1}{2}\sum_f F_f^d(\delta\hat{u}_i)$$
$$= -\sum_f \overline{p^n} n_i A - \sum_f F_f^c\left(u_i^n, U^n\right) - \sum_f F_f^d\left(u_i^n\right), \tag{12}$$

$$u_i^* - \hat{u}_i = \frac{\Delta t}{\mathrm{Vol}}\sum_f \overline{p^n} n_i A, \tag{13}$$

$$U^* = \overline{u_i^* n_i}, \tag{14}$$

$$\sum_f \frac{\partial p^{n+1}}{\partial x_i} n_i A = \frac{1}{\Delta t}\sum_f U^* A, \tag{15}$$

$$u_i^{n+1} - u_i^* = -\frac{\Delta t}{\mathrm{Vol}}\sum_f \overline{p^{n+1}} n_i A, \tag{16}$$

$$U^{n+1} - U^* = -\Delta t\frac{\partial p^{n+1}}{\partial x_i} n_i, \tag{17}$$

where $\delta\hat{u}_i = \hat{u}_i - u_i^n$, Δt is the computational time step, and Vol is the cell volume. In all equations, summation is implied over repeated indices. The components of the pressure gradient required by Eqs. (15) and (17) are approximated using the same "auxiliary node" technique developed for the viscous flux approximation.

In the present work, the sparse system resulting from Eq. (12) was solved iteratively using coupled Gauss–Seidel iteration. After every iteration, the required gradients involved in the lagged terms were updated. The stiffness was not found to be excessive, reducing the residual by six orders of magnitude in 10 to 15 iterations without the use of multigrid or more complex solvers, even for the largest problems considered. For the Poisson system of Eq. (15), multigrid was found to significantly reduce the computation time, particularly for the largest problems considered. Boundary conditions were handled as described by Kim and Choi.

3.3.2. *Semi-implicit Approach*

To remove the implicit coupling between velocity components and allow the application of approximate factorization to advance the velocity field, a semi-implicit approach was also considered. Applying second-order Adams–Bashforth to the convective terms and the lagged part of the viscous terms, Eq. (12) for the velocity increment, δu_i, becomes

$$\begin{aligned}\frac{\delta \hat{u}_i}{\Delta t}\text{Vol} + \frac{1}{2}\sum_f F_f^{d,Active}(\delta \hat{u}_i)\\ = -\sum_f \overline{p^n} n_i A - \sum_f F_f^{d,Active}\left(u_i^n\right) - \frac{3}{2}\left(\sum_f F_f^c\left(u_i^n, U^n\right) + \sum_f F_f^{d,Lagged}\left(u_i^n\right)\right)\\ + \frac{1}{2}\left(\sum_f F_f^c\left(u_i^{n-1}, U^{n-1}\right) + \sum_f F_f^{d,Lagged}\left(u_i^{n-1}\right)\right).\end{aligned} \tag{18}$$

The remainder of the equations are identical to those described by Eqs. (13) through (17).

As discussed previously, this semi-implicit approach has the disadvantage of introducing a Courant–Freidrichs–Lewy limit on numerical time step of the form $\text{CFL} = \max(|u_i|\Delta t/\Delta x_i) \approx 1$. On Cartesian meshes with highly refined cells, this can become excessively limiting. On the other hand, the solution of Eq. (18) can be significantly less costly than solving Eq. (12) in terms of both memory and computation. For the overall algorithm (including multigrid solution of the pressure equation, which is the same in both cases), the semi-implicit approach was found to be about three times as fast per time step when compared to the fully implicit approach. For relatively small two-dimensional problems, the increased memory requirement of the fully implicit approach was not an issue, although it may become an important consideration for large, three-dimensional problems.

3.4. Approximate Factorization

When the fractional step method is used to solve the unsteady Navier–Stokes equations on structured grids, it is common to use approximate factorization to advance the velocity field [7]. Approximate factorization reduces the problem of inverting the large sparse matrix associated with the discretized momentum equations to one of inverting a series of tridiagonal matrices. These smaller one-dimensional systems can be directly inverted using the tridiagonal matrix algorithm, resulting in a significant reduction in both computing cost and memory requirement.

On a mesh of Cartesian cells with local refinement, the active part of the diffusive flux discretization can be considered either an x- or y-discrete operator, depending on the orientation of the associated face. Consequently, when the semi-implicit approach to time advancement is used, the discretization equation in each cell can readily be written in a form suitable for approximate factorization. Unfortunately the resulting one-dimensional systems will not be tridiagonal because the local refinement and coarsening introduces multiple connections along any given direction. When the Cartesian cells are properly ordered, however, these one-dimensional systems can be efficiently and directly inverted using a Gaussian elimination technique. This technique is now presented in detail.

Figure 4 shows a typical "linkage" of Cartesian cells, considering just their mutual coupling through the discrete x-operators. Any given cell is also a member of a second linkage extending in the y-direction. The linkage represents the coupled one-dimensional system that must be inverted after approximate factorization has been applied to the semi-implicit equation for the velocity increment, Eq. (18). The total width (in the y-direction in this case) of the linkage is determined by the maximum width of any cell in the linkage, and its length (x-direction in this case) spans the entire domain.

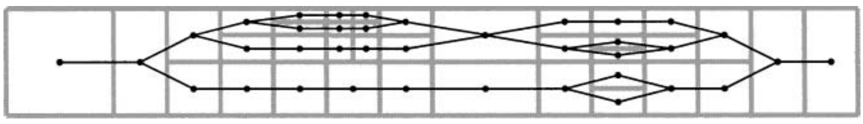

FIG. 4. A typical "linkage" of Cartesian cells, considering just their mutual coupling through the discrete x-operators.

The method of Gaussian elimination is a two-phased approach for inverting a coupled linear system of equations. In phase one, the first equation is used to eliminate the dependence on the first unknown from all other equations. This is repeated for each unknown, eventually producing an upper-triangular matrix. A second phase of back-substitution is then performed to compute the solution. The method of Gaussian elimination is rarely applied to even medium-sized sparse systems because the upper-triangular matrix can quickly become nearly full, requiring substantial memory and computation time and making the solution prone to numerical error associated with compounded machine round-off.

When the coupled system is that resulting from a linkage of Cartesian cells, however, and the cells are ordered according to width from narrowest to widest, Gaussian elimination can very efficiently invert the resulting system. Because of the ordered connectivity of cells, the expanding of the coefficient matrix after successive eliminations does not occur. For clarity, Fig. 5 illustrates a portion of the Gaussian elimination process using the typical linkage of cells introduced previously.

Finally we point out that, while necessary for the efficient implementation of approximate factorization, this Gaussian elimination technique can also be used as a line-smoother for

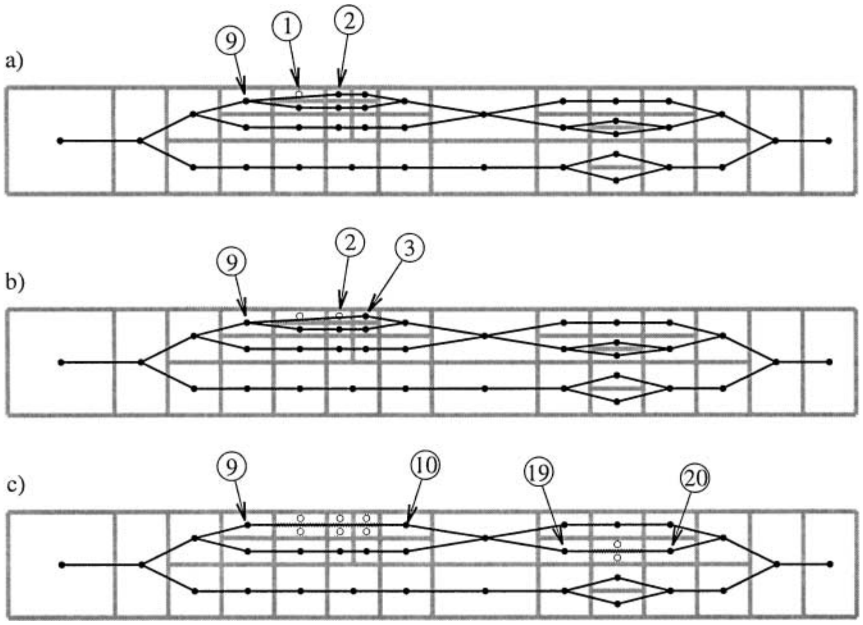

FIG. 5. Illustration of the Gaussian elimination process using a typical "linkage" of Cartesian cells. All cells in the linkage are sorted from smallest to largest width (Δy in this case). (a) The dependence on cell 1 is eliminated from its neighbours, cells 9 and 2. Cell 9's neighbour reference to cell 1 is changed to reference cell 2, and the coefficients are modified appropriately. Similarly, cell 2's neighbour reference to cell 1 is changed to reference cell 9. (b) The dependence on cell 2 is eliminated from its current neighbours (cells 9 and 3). (c) Remaining neighbour connectivity after the elimination of several more cells.

the iterative multigrid solution of the pressure equation, and even the momentum equations when solved iteratively.

3.5. Multigrid Solution of Pressure Poisson Equation

In the present method, the pressure Poisson equation is solved using additive correction multigrid (ACM) [17]. The performance of ACM, however, can depend critically on the cell agglomeration routine used to define the coarse meshes. As discussed by Raw [18], optimal coarsening should be performed in the direction of greatest coefficient strength. Raw goes on to develop a coarsening algorithm based on this principle. Because the algorithm is developed for fully unstructured meshes, it requires ad hoc modifications to prevent the coarse cells from becoming excessively irregular, or from having an excessively large number of neighbours. That is, a certain degree of smoothness is desired in the coarse cells.

For the present case of Cartesian cell meshes with anisotropic local refinement, a natural coarse mesh consists simply of coarser Cartesian cells. By requiring that the coarse cells at all multigrid levels also be Cartesian, we avoid the problems associated with irregular coarse cells. We also note that, for the solution of the pressure Poisson equation, the direction of greatest coefficient strength can be interpreted geometrically as the coarsening direction that most reduces cell anisotropy.

In the present work, the coarse Cartesian mesh is stored using the same unstructured data format and cell indices introduced for the fine mesh. The coarse mesh is defined as the largest set of nonoverlapping cells for which each coarse cell's level indices $[l_I, l_J]$ are less than (i.e., coarser) or equal to all the "target coarse cell indices" of their contained fine cells. Here we have introduced capital letters to distinguish coarse cells from fine cells. The "target coarse cell indices" for any fine cell are calculated using the following algorithm.

```
getTargetCoarseCellIndices
1. Determine the maximum (i.e., finest) refinement level of any neighbour
   $l_{i,max} = 0, l_{j,max} = 0$
   for all cell neighbours $c_{nb}$
       $l_{i,max} = max(l_{i,max}, l_{i,nb})$
       $l_{j,max} = max(l_{j,max}, l_{j,nb})$
   end for
2. Set the target coarse cell's x-indices $l_I$ and $I$
   if $(l_i \geq l_{i,max})$ AND $(\Delta x < \alpha \Delta y)$ AND $(l_i > 0)$
       $l_I = l_i - 1, I = i/2$
   else
       $l_I = l_i, I = i$
   endif
3. Set the target coarse cell's y-indices $l_J$ and $J$
   if $(l_j \geq l_{j,max})$ AND $(\Delta y < \alpha \Delta x)$ AND $(l_j > 0)$
       $l_J = l_j - 1, j = j/2$
   else
       $l_J = l_j, J = j$
   endif
end getTargetCoarseCellIndices
```

In the above algorithm, $\alpha \geq 1$ is an adjustable parameter that prevents the definition of the

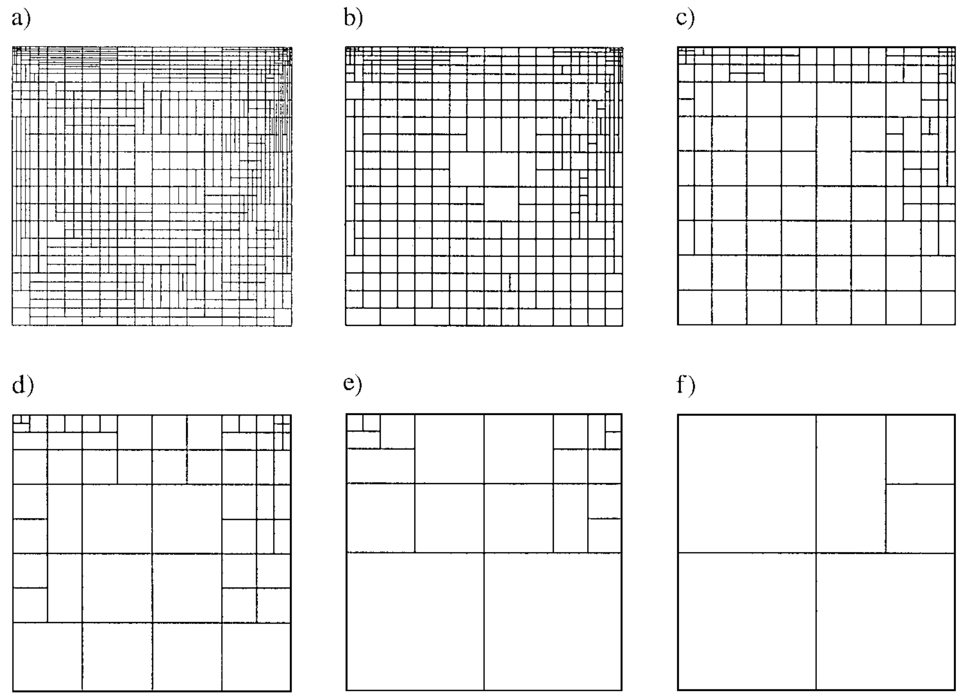

FIG. 6. Fine mesh and multigrid coarse mesh levels determined by applying the coarsening strategy. (a) Fine grid (823 cells). (b) Coarse grid level 1 (374 cells). (c) Coarse grid level 2 (143 cells). (d) Coarse grid level 3 (58 cells). (e) Coarse grid level 4 (21 cells). (f) Coarse grid level 5 (six cells).

coarse grid in a manner that increases the cell anisotropy excessively; $\alpha = 1.5$ was used throughout the present work.

Figure 6 shows a typical fine mesh and five multigrid coarse mesh levels determined by applying this coarsening strategy. These particular meshes are taken from the two-dimensional lid-driven cavity flow problem, presented in Section 5. The average coarsening ratio is about 2.6 fine cells per coarse cell. This relatively low coarsening ratio increases the memory overhead associated with the multigrid implementation, but gives good convergence results even with relatively weak smoothers (e.g., point-iterative Gauss–Seidel).

3.6. Time-Accurate Interpolation

After any cell is refined or coarsened, it is necessary to interpolate the cell and face data to the new data location(s). In the present implementation, a second-order interpolation of the data is performed to any new locations after each refinement or coarsening. This corresponds to averaging the data between two cells that are coarsened into one, or extrapolating the data using the cell gradient when one cell is refined into two. The face-normal mass-conserving velocities are handled similarly. Although higher order derivatives are available, the interpolation is limited to second order because of desirable conservation properties: namely that the new face-based velocities will still satisfy discrete conservation of mass, and the new cell-centered velocities will discretely conserve the local and global momentum. For example, the integrated u-momentum,

$$M_u = \int_V u\, dV = \sum u_c V_c, \tag{19}$$

remains constant over any consistent volume V before and after refinement or coarsening. These conservation properties ensure that neither mass nor momentum is created or destroyed in the adaptation process, thus allowing the time-accurate advancement of the solution.

We note that, when the two-level semi-implicit scheme is used, data from both the current and previous time levels are interpolated as described above. This ensures the spatial correspondence of the data at both time levels.

4. GRID ADAPTATION CRITERION

The goal of this section is to develop expressions for the target anisotropic cell dimensions, Δx_{target} and Δy_{target}. Ideally, these expressions should be simple, explicit functions of the variables and/or derivatives at the cell centroids,

$$\Delta x_{target} = f\left(u_c, v_c, \left.\frac{\partial u}{\partial x}\right|_c, \left.\frac{\partial^2 u}{\partial x^2}\right|_c, \ldots\right), \tag{20}$$

$$\Delta y_{target} = g\left(u_c, v_c, \left.\frac{\partial u}{\partial x}\right|_c, \left.\frac{\partial^2 u}{\partial x^2}\right|_c, \ldots\right). \tag{21}$$

The efficient adaptation of the mesh can then by accomplished by simply comparing the actual cell dimensions to the target dimensions calculated using Eqs. (20) and (21), and then refining or coarsening appropriately.

Recalling the three ingredients of any adaptive refinement scheme [9],

1. an error indicator,
2. an optimal mesh criterion, and
3. an algorithm to refine and coarsen the mesh,

this section develops the first two (the third has already been described in Section 2) to derive analytic expressions for the optimal local anisotropic cell dimensions.

4.1. Error Indicator

When the finite volume method is applied to the Navier–Stokes equations, integrals of the velocity over the Cartesian control volume Ω are approximated by taking the value at the centroid and multiplying by the cell volume. For example, for the u-velocity

$$\int_\Omega u\, d\Omega = u_c \Delta x \Delta y + \varepsilon_u, \tag{22}$$

where u_c is the value at the cell centroid (x_c, y_c), Δx and Δy are the Cartesian cell dimensions, and ε_u is the error associated with the approximation. If u is assumed to be smoothly varying throughout the cell, the error can be estimated by integrating the two-dimensional Taylor series expansion of u about the centroid over the Cartesian control volume. Retaining only the lowest order terms, the error expression becomes

$$\varepsilon_u = \frac{\Delta x^3 \Delta y}{24}\frac{\partial^2 u}{\partial x^2} + \frac{\Delta x \Delta y^3}{24}\frac{\partial^2 u}{\partial y^2}, \tag{23}$$

where $\frac{\partial^2 u}{\partial x^2}$ and $\frac{\partial^2 u}{\partial y^2}$ are evaluated at the cell centroid.

4.2. Optimal Mesh Criterion

We consider the optimal mesh to be the smallest mesh (i.e., mesh with the fewest cells) for which the error associated with each cell satisfies

$$\varepsilon^2 \leq c^2, \tag{24}$$

where ε is some measure of the error (squared to ensure positivity), and c is a specified error tolerance. In the present work, Eq. (23) is used as the basis for the error expression. More specifically, we define the optimal mesh as the smallest mesh that satisfies the following criteria:

$$\varepsilon^2 = \varepsilon_u^2 + \varepsilon_v^2 \leq c^2. \tag{25}$$

Substituting Eq. (23) and a similar expression for ε_v into Eq. (25) yields the following relationship:

$$\left(\frac{\Delta x^3 \Delta y}{24}\frac{\partial^2 u}{\partial x^2} + \frac{\Delta x \Delta y^3}{24}\frac{\partial^2 u}{\partial y^2}\right)^2 + \left(\frac{\Delta x^3 \Delta y}{24}\frac{\partial^2 v}{\partial x^2} + \frac{\Delta x \Delta y^3}{24}\frac{\partial^2 v}{\partial y^2}\right)^2 \leq c^2. \tag{26}$$

Following Simpson [19], this global optimization problem is equivalent to a local maximization problem: maximize the cell area subject to the constraint defined by Eq. (26). Stated differently, the optimal combination of $(\Delta x, \Delta y)$ that locally satisfies Eq. (26) is simply the combination for which the cell area ($A = \Delta x \Delta y$) is maximal.

Using Eq. (26) and the maximal area concept, it is possible to develop expressions for the target Cartesian cell dimensions based on the second velocity derivatives at the cell centroid. Unfortunately, the resulting expressions are cumbersome, violating our requirement for simple, explicit expressions. We now make some simplifications to Eq. (26) that produce simpler explicit expressions for the target cell dimensions, but compromise our definition of optimality slightly.

First, note that Eq. (26) can be rewritten as

$$\left(\frac{\Delta x^3 \Delta y}{24} F_{xx} + \frac{\Delta x \Delta y^3}{24} F_{yy}\right)^2 + \gamma \leq c^2, \tag{27}$$

where F_{xx} and F_{yy} are the magnitudes of the cell centroid derivatives, defined as

$$F_{xx} = \sqrt{\left(\frac{\partial^2 u}{\partial x^2}\right)^2 + \left(\frac{\partial^2 v}{\partial x^2}\right)^2}, \tag{28}$$

$$F_{yy} = \sqrt{\left(\frac{\partial^2 u}{\partial y^2}\right)^2 + \left(\frac{\partial^2 v}{\partial y^2}\right)^2}, \tag{29}$$

and γ is the expression

$$\gamma = \frac{\Delta x^4 \Delta y^4}{288}\left(\frac{\partial^2 u}{\partial x^2}\frac{\partial^2 u}{\partial y^2} + \frac{\partial^2 v}{\partial x^2}\frac{\partial^2 v}{\partial y^2} - F_{xx} F_{yy}\right). \tag{30}$$

By analyzing Eq. (30), it is straightforward to prove that $\gamma \le 0$ for all possible combinations of second derivatives. Consequently, we introduce the modified criterion

$$\left(\frac{\Delta x^3 \Delta y}{24} F_{xx} + \frac{\Delta x \Delta y^3}{24} F_{yy}\right)^2 \le c^2, \tag{31}$$

or, taking the square root of both sides,

$$\frac{\Delta x^3 \Delta y}{24} F_{xx} + \frac{\Delta x \Delta y^3}{24} F_{yy} \le c. \tag{32}$$

Because $\gamma \le 0$, any mesh that satisfies Eq. (32) in every cell is guaranteed to satisfy Eq. (26) in every cell. Although the smallest mesh that satisfies Eq. (32) will not necessarily correspond to the optimal mesh as defined earlier, at least the error is guaranteed to be below the specified error tolerance c. Further, the expressions for Δx and Δy turn out to be much simpler. The cell dimensions that maximize the area, $A = \Delta x \Delta y$, subject to the constraint of Eq. (32) are

$$\Delta x_{target} = \left(\frac{144 c^2 F_{yy}}{F_{xx}^3}\right)^{\frac{1}{8}}, \tag{33}$$

$$\Delta y_{target} = \left(\frac{144 c^2 F_{xx}}{F_{yy}^3}\right)^{\frac{1}{8}}. \tag{34}$$

In Eqs. (33) and (34), the subscript "target" has been introduced as a reminder that the simplifications involved in these expressions do not necessarily result in an optimal mesh. To reiterate, however, the error as defined in Eq. (23) will still be limited by the user-specified error tolerance c.

4.3. Three-Dimensional Expressions

In three dimensions, the expression corresponding to the error criteria of Eq. (32) is

$$\frac{\Delta x^3 \Delta y \Delta z}{24} F_{xx} + \frac{\Delta x \Delta y^3 \Delta z}{24} F_{yy} + \frac{\Delta x \Delta y \Delta z^3}{24} F_{zz} \le c, \tag{35}$$

where F_{xx}, F_{yy}, and F_{zz} are the magnitudes of the second derivatives calculated at the cell centroids,

$$F_{xx} = \sqrt{\left(\frac{\partial^2 u}{\partial x^2}\right)^2 + \left(\frac{\partial^2 v}{\partial x^2}\right)^2 + \left(\frac{\partial^2 w}{\partial x^2}\right)^2}, \tag{36}$$

$$F_{yy} = \sqrt{\left(\frac{\partial^2 u}{\partial y^2}\right)^2 + \left(\frac{\partial^2 v}{\partial y^2}\right)^2 + \left(\frac{\partial^2 w}{\partial y^2}\right)^2}, \tag{37}$$

$$F_{zz} = \sqrt{\left(\frac{\partial^2 u}{\partial z^2}\right)^2 + \left(\frac{\partial^2 v}{\partial z^2}\right)^2 + \left(\frac{\partial^2 w}{\partial z^2}\right)^2}. \tag{38}$$

The target cell dimensions maximize the cell volume, $V = \Delta x \Delta y \Delta z$, subject to the constraint defined by Eq. (35). This problem is most easily solved using the method of

Lagrange multipliers [20], yielding the target mesh dimensions

$$\Delta x_{target} = \left(\frac{64c^2 F_{yy} F_{zz}}{F_{xx}^4} \right)^{\frac{1}{10}}, \tag{39}$$

$$\Delta y_{target} = \left(\frac{64c^2 F_{xx} F_{zz}}{F_{yy}^4} \right)^{\frac{1}{10}}, \tag{40}$$

$$\Delta z_{target} = \left(\frac{64c^2 F_{xx} F_{yy}}{F_{zz}^4} \right)^{\frac{1}{10}}. \tag{41}$$

5. VALIDATION

To validate the grid adaptation criteria along with the overall Cartesian grid method, some results from the following benchmark flow problems are now presented: the 2D and 3D lid-driven cavity flows and the flow around a circular cylinder. In this latter case, an immersed boundary method is used to handle the embedded cylinder boundary.

5.1. 2D Lid-Driven Cavity

The two-dimensional lid-driven cavity flow is a simple viscous incompressible flow that has many features suitable for testing the performance of an anisotropic mesh adaptation strategy, including primary and secondary vortices, wall boundary layers, and flow separation and reattachment. The flow was studied numerically by Ghia *et al.* [21], who reported accurate solutions to the steady flow solution over a range of Reynolds numbers ($\text{Re} = UL/\nu$, where U is the constant lid velocity and L the cavity dimension).

To test the adaptation strategy of the present contribution, the cavity flow problem was solved on a variety of Cartesian meshes, both with and without adaptation. The flow was started from rest on a uniform Cartesian mesh and integrated ahead in time until a steady solution was reached. For the cases involving adaptation, following each time step the local target cell dimensions were calculated and cells refined or coarsened appropriately. A simple feedback mechanism was used to adjust the user-specified error tolerance c throughout the calculation to keep the number of cells approximately constant.

Figure 7 qualitatively illustrates the effect of the anisotropic adaptation on the mesh by comparing a uniform mesh to an adapted mesh with approximately the same number of cells.

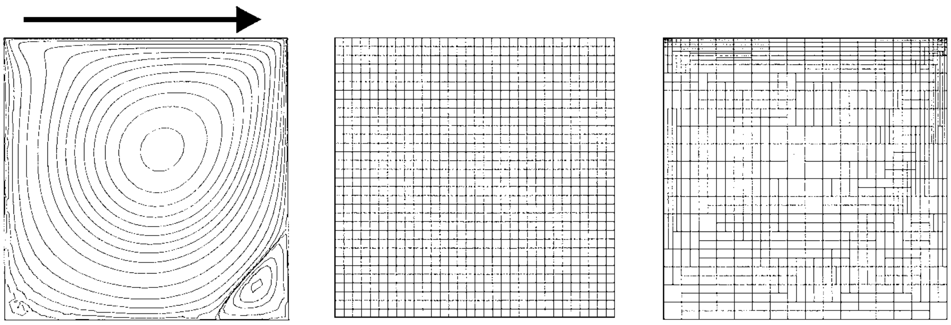

FIG. 7. Streamlines, uniform mesh, and adapted mesh for the steady-state solution to the 2D lid-driven cavity flow at Reynolds number $\text{Re} = 400$. Both meshes shown have approximately the same number of cells, $N \approx 1000$.

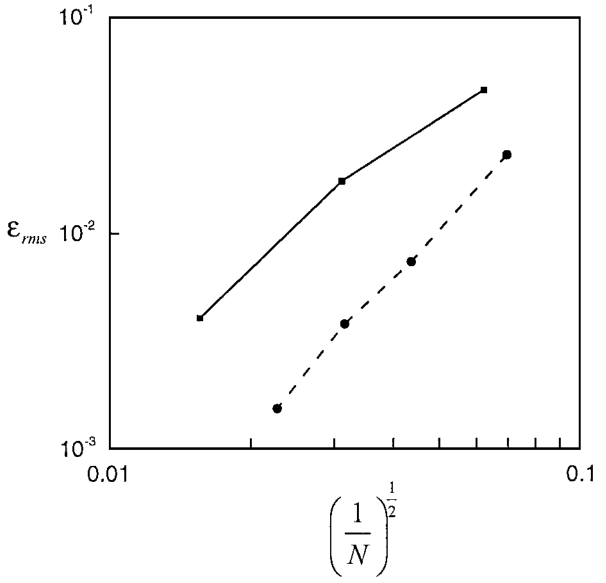

FIG. 8. Rms solution error as defined by Eq. (42) against average cell dimension for the 2D lid-driven cavity flow problem at Reynolds number Re = 400: ——— uniform Cartesian mesh; - - - adapted Cartesian mesh.

Plots comparing centerline velocities are not given because they are nearly identical to the results of Ghia *et al.*, particularly on the finer meshes considered. To quantify the difference in solution error between the various meshes, the following rms velocity error is defined,

$$\varepsilon_{rms} = \sqrt{\frac{\sum_{i=1}^{15}(u(x_i, y_i) - u_{Ghia}(x_i, y_i))^2}{15}}, \tag{42}$$

where (x_i, y_i) are the 15 locations along the vertical centerline reported by Ghia *et al.* (excluding the boundary results at $y = 0$ and $y = 1$), u are the steady velocity values from the present calculations interpolated to the Ghia points, and u_{Ghia} are the velocity values reported by Ghia *et al.* from their 129×129 grid [21]. Figure 8 compares this rms error on the uniform and adapted meshes plotted against the average cell dimension, where N is the total number of cells.

Throughout the entire range of problem sizes investigated, the solution on the adapted mesh is significantly more accurate than the solution on the unadapted mesh. For example, at a problem size of approximately $N = 1000$ cells, mesh adaptation results in a reduction in the rms solution error by a factor of about 5. In addition, the rate of error reduction (i.e., slope) for the adapted case is approximately 2.4, indicating a slight superconvergence as might be expected when (some of) the truncation errors associated with the present second-order scheme are minimized.

5.2. 3D Lid-Driven Cavity

The 3D lid-driven cavity involves the flow in a cubical enclosure with lid moving at a constant velocity $U = 1$. The presence of the two additional walls significantly complicates the flow by introducing secondary mean flows due to the interaction of the rotating fluid and the no-slip condition at these walls. A 3D version of the present adaptive Cartesian method

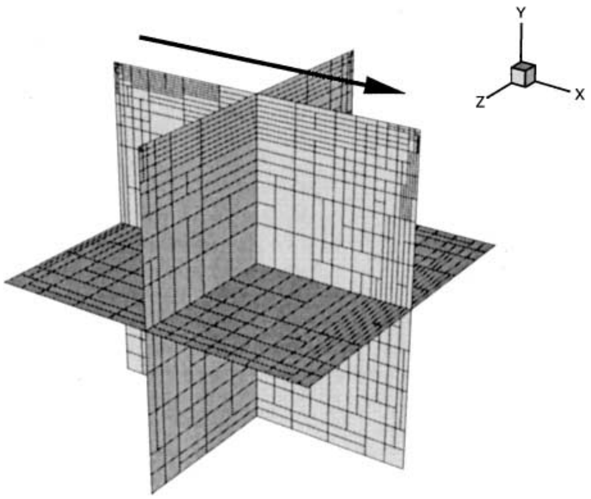

FIG. 9. Anisotropic Cartesian mesh shown on three orthogonal planes through the center of the 3D lid-driven cavity at Re $= UL/\nu = 400$. The arrow indicates the location and direction of the driven lid.

was used to solve this problem. As with the 2D cavity, the flow was started from rest and integrated ahead in time until a steady solution was reached. Adaptation was performed after every time step.

Figure 9 attempts to illustrate the 3D anisotropic mesh by viewing three orthogonal planes passing through the center of the cavity. Note the high degree of anisotropy in the cells near the driven lid. Figure 10 shows the velocity vectors on the plane of symmetry. The significant reduction in the total number of cells required to achieve a given level of accuracy is illustrated by Fig. 11, which compares the calculated velocity profiles along the vertical centerline from four different simulations. When compared to the grid-independent result, the adapted Cartesian grid solution with about 2000 cells is slightly superior to the uniform grid solution with over 30,000 cells.

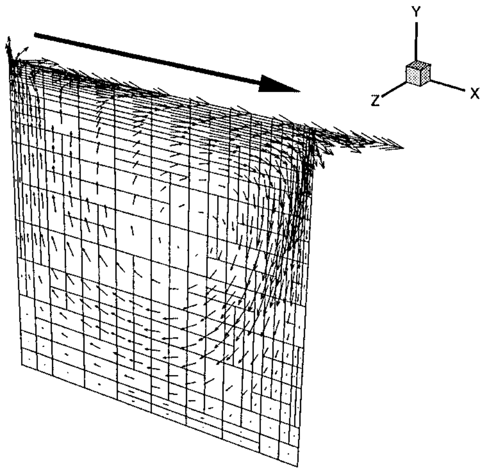

FIG. 10. Anisotropic Cartesian mesh and velocity vectors on the center plane of symmetry of the 3D lid-driven cavity at Re = 400.

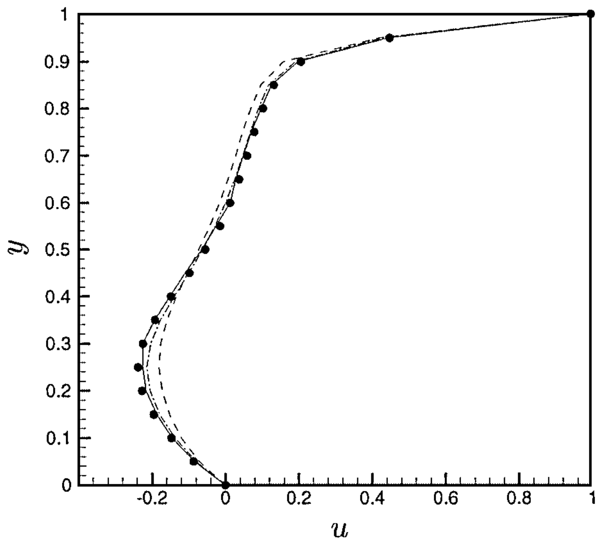

FIG. 11. Comparison of calculated u-velocity along the vertical centerline of the 3D lid-driven cavity at Re = 400: ——— adapted anisotropic Cartesian grid (2000 cells); --- uniform grid ($16^3 = 4096$ cells); – · – uniform grid ($32^3 = 32768$ cells); ● grid-independent result.

5.3. Flow Around a Circular Cylinder

Both of the previous flows have involved simple Cartesian geometries and viscous boundary layers aligned with the principal coordinate directions. In this last case considered, a body force technique is used to embed a complex boundary in the Cartesian grid and simulate the time-dependent flow around a circular cylinder. Above a critical Reynolds number of about $\mathrm{Re} = U_\infty D/\nu > 35$, the flow around a circular cylinder is unsteady and periodic, producing the well-known von Karmann vortex street. The flow has been extensively studied both experimentally and numerically, and thus serves as a suitable benchmark unsteady incompressible flow to test the time-accuracy of the transient mesh adaptation.

The concept of adding body forces to the momentum equations to simulate the presence of boundaries dates back to the pioneering work of Peskin, who investigated the blood flow in a beating heart [22, 23]. His method has come to be known as the immersed boundary method (IMBM). IMBM does not require the correspondence of grid lines with the boundaries, thus permitting the use of structured Cartesian meshes to simulate flows involving complex and moving boundaries. Recently, Roma *et al.* [24] combined the immersed boundary method with an embedded grid technique to produce a time-adaptive method for moving boundary problems, with promising two-dimensional results. The optimal selection of embedded nonoverlapping grids, however, remains a complex problem. For example, the two-dimensional algorithm used by Roma *et al.* “combines elements of both computer vision and pattern recognition theory” [24], and no comment is made on its suitability in three dimensions. Lai and Peskin recently applied the immersed boundary method to solve the cylinder flow problem, although only uniform Cartesian grids were considered [25].

For the present cylinder flow simulation, we use a simplified form of IMBM suitable for flows where the coupling between moving boundaries and the flow is only one-way (i.e., the boundary is known as a function of space and time) [26, 27]. In practice, this technique is

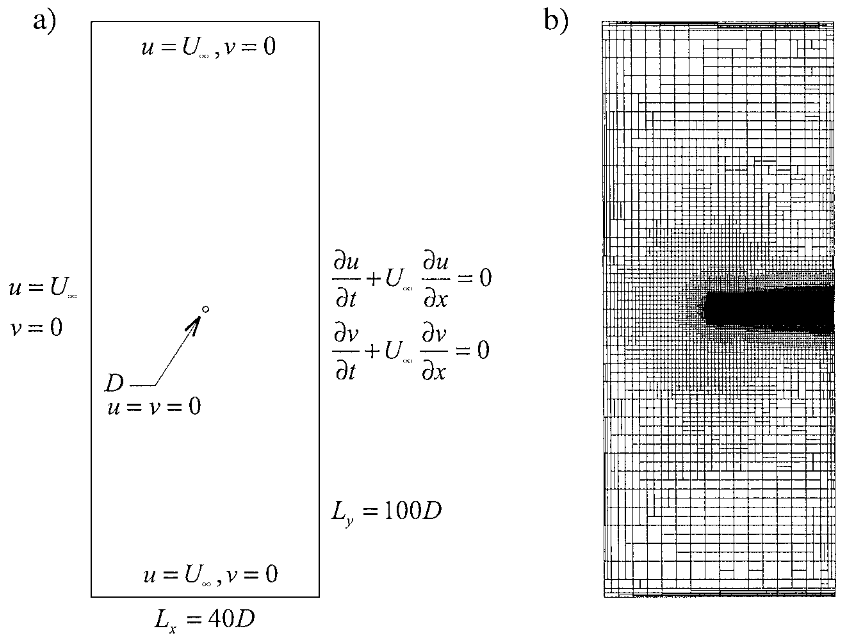
a)
$u = U_\infty, v = 0$
$u = U_\infty$
$v = 0$
D
$u = v = 0$
$\frac{\partial u}{\partial t} + U_\infty \frac{\partial u}{\partial x} = 0$
$\frac{\partial v}{\partial t} + U_\infty \frac{\partial v}{\partial x} = 0$
$L_y = 100D$
$u = U_\infty, v = 0$
$L_x = 40D$
b)

FIG. 12. (a) Geometry and boundary conditions for the flow around a circular cylinder. (b) Complete view of a typical adapted Cartesian mesh.

implemented by simply linearly interpolating the velocity field in the fluid control volumes immediately adjacent to the immersed boundary.

Figure 12a shows the problem geometry and boundary conditions. To facilitate a comparison of results, the geometry is identical to that reported by Kim and Choi [8]. We note that no modification has been made to the mesh adaptation scheme to preferentially refine the mesh near the cylinder boundaries. Refinement near the cylinder boundaries occurs naturally because of the effect of the cylinder on the flow solution. The flow was started impulsively from a zero velocity field. Above the critical Reynolds number, vortex shedding was found to initiate spontaneously and did not require the perturbation of the flow field. This was presumably due to slight asymmetries in the adapted mesh, which can occur from time to time, even when the flow is theoretically symmetric. The flow was computed until the frequency of vortex shedding remained constant. Three Reynolds numbers were investigated, Re $=$ 80, 100, and 120. In all cases, the fully implicit discretization of convective terms proposed by Kim and Choi was used, and the time step was adjusted to maintain CFL $= \max(u_i \Delta t / \Delta x_i) \approx 1$.

Figure 12b shows a complete view of a typical adapted Cartesian mesh. In this case, the mesh has approximately 12,000 cells, corresponding to the adaptation criterion $c = 5 \times 10^{-5}$. Note that the mesh adaptation scheme has produced a few highly anisotropic cells at the north and south boundaries. This is a result of the Dirichlet boundary condition used at these locations ($u = U_\infty$, $v = 0$), which produces a slight shear layer. Refinement at these locations could be avoided by either modifying this boundary condition, or making the adaptation criterion c of Eqs. (33) and (34) a function of space, rather than simply a global constant.

Figure 13 shows the Cartesian mesh and corresponding contours of pressure around the cylinder and in the wake region at one instant in time. Although the mesh adaptation is

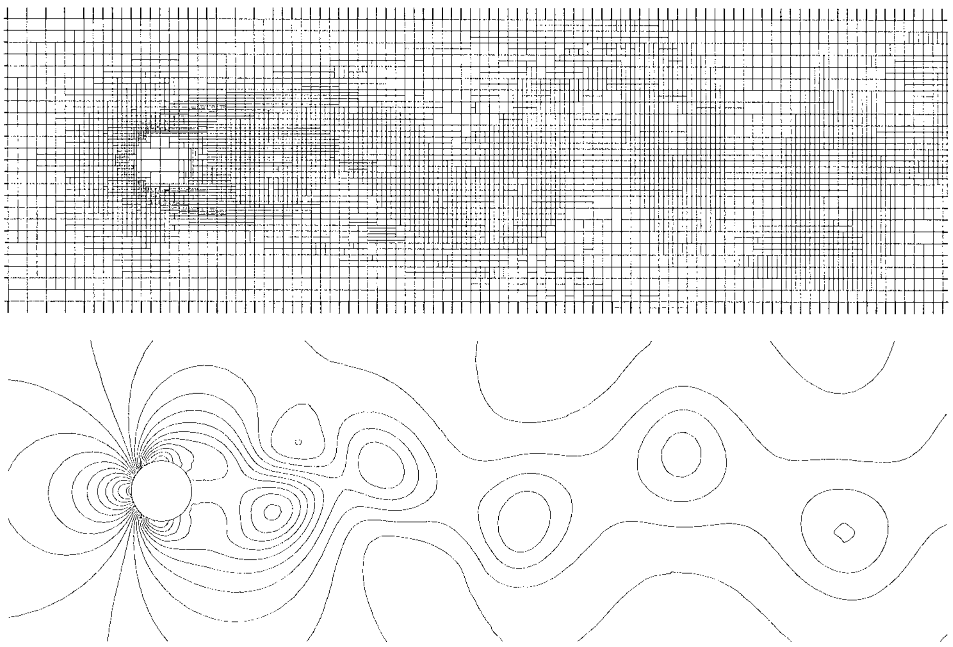

FIG. 13. Detail of the Cartesian mesh and corresponding contours of pressure in the wake region of the flow around a circular cylinder at Re $= 100$ (contour spacing $\Delta p = 0.05$).

anisotropic, only the boundary layers near the cylinder and flow in the very near wake contain significant cell anisotropy at these relatively low Reynolds numbers (see Fig. 14).

Figure 15 shows the calculated nondimensional frequency of vortex shedding, or Strouhal number, as a function of Reynolds number. The results are in excellent agreement with the calculation of Kim and Choi [8] and the correlation of Williamson [28].

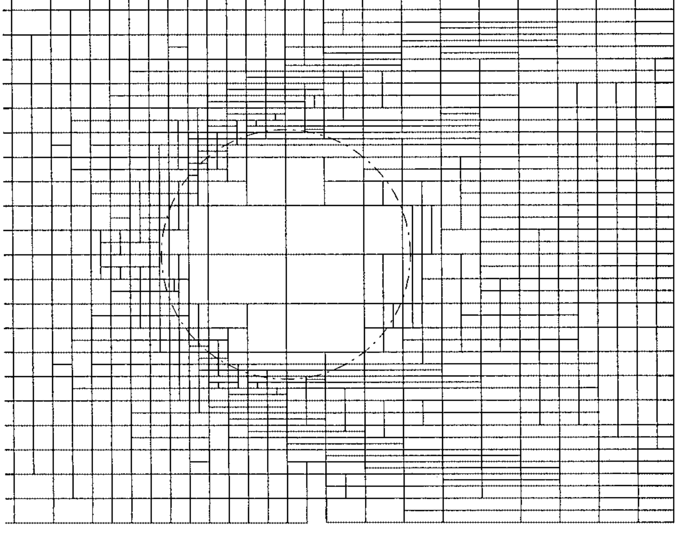

FIG. 14. Detail of the Cartesian mesh near the cylinder for Re $= 100$: – · – immersed boundary.

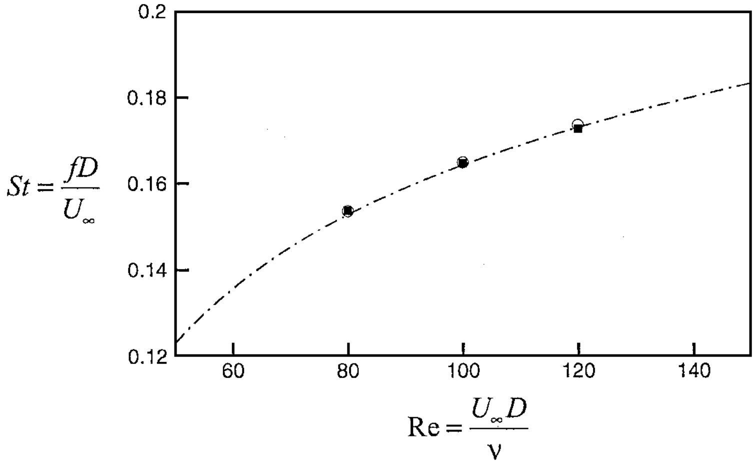

FIG. 15. Variation of Strouhal number with Reynolds number: ■ present study; ○ Kim and Choi [8]; – · – correlation of Williamson [28].

To demonstrate the approach to mesh independence, Fig. 16 plots the error in calculated Strouhal number (relative to the correlation of Williamson) for all grid sizes and Reynolds numbers investigated. This error is plotted against the nominal Cartesian cell dimension, defined in terms of the total number of cells, N, and the domain dimensions L_x and L_y. Although somewhat scattered, the rate of error reduction appears to be consistently steeper than second order.

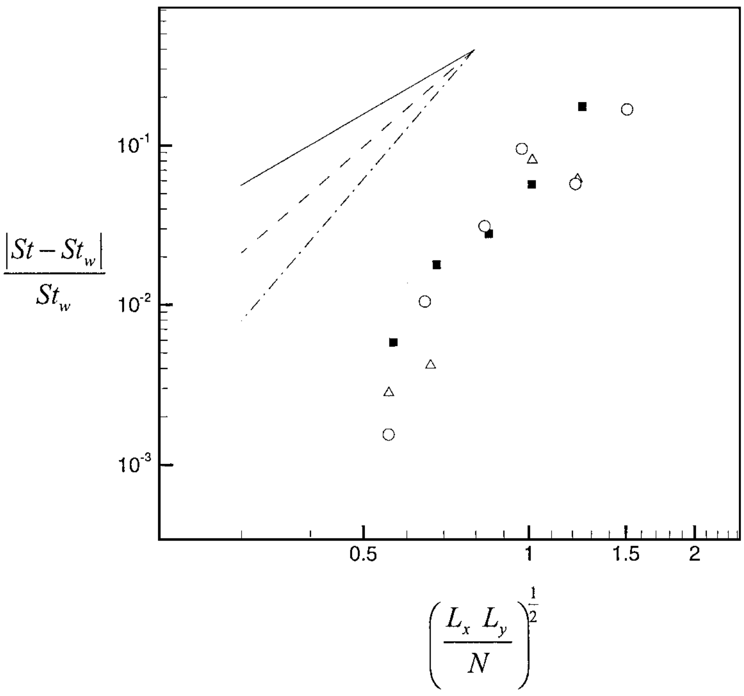

FIG. 16. Strouhal number error relative to the correlation of Williamson St_w [28] vs nominal Cartesian cell dimension (N = total number of cells, L_x, L_y = domain dimensions): ■ Re = 80; △ Re = 100; ○ Re = 120; ——— slope 2; - - - slope 3; – · – slope 4.

6. SUMMARY

A Cartesian mesh method with local anisotropic refinement and coarsening has been developed for simulating time-dependent incompressible flows. Algorithms have been described for the time-accurate transient anisotropic refinement and coarsening of the cells. We have shown how the method of approximate factorization, common to structured grid methods, can be applied to advance the velocity field during the solution of each time step. An efficient coarse grid selection algorithm has been described for the multigrid solution of the pressure Poisson equation. Simple analytic expressions for the optimal anisotropic mesh dimensions have been derived based on limiting the local velocity interpolation error. Finally, the overall method has been validated by solving three benchmark flows, all with excellent results.

ACKNOWLEDGMENTS

The authors are grateful to the Natural Sciences and Engineering Research Council of Canada (NSERC) for the financial support of this work. We also acknowledge the referees, whose helpful suggestions resulted in the improvement of the manuscript.

REFERENCES

1. W. J. Coirier and K. G. Powell, An accuracy assessment of Cartesian-mesh approaches for the Euler equations, *J. Comput. Phys.* **117**, 121 (1995).
2. M. Aftosmis, J. Melton, and M. Berger, Robust and efficient Cartesian mesh generation for component-based geometry, *AIAA J.* **36**(6), 952 (1998).
3. M. J. Aftosmis, M. J. Berger, and G. Adomavicius, A parallel multilevel method for adaptively refined Cartesian grids with embedded boundaries, in *38th Aerospace Sciences Meeting and Exhibit, January 10–13, 2000, Reno, NV* (AIAA 2000-0808).
4. M. Delanaye, M. J. Aftosmis, M. J. Berger, Y. Liu, and T. H. Pulliam, Automatic hybrid-Cartesian grid generation for high-Reynolds number flows around complex geometries, in *37th AIAA Aerospace Sciences Meeting and Exhibit, January 11–14, 1999, Reno, NV* (AIAA 99-0777).
5. W. J. Coirier and K. G. Powell, Solution-adaptive Cartesian cell approach for viscous and inviscid flows, *AIAA J.* **34**(5), 938 (1996).
6. M. J. Berger and M. J. Aftosmis, Aspect (and aspect ratios) of Cartesian mesh methods, in *Proceedings of the 16th International Conf. on Num. Meth. in Fluid Dynamics, 6–10 July, 1998, Arcachon, France*.
7. J. Kim and P. Moin, Application of a fractional-step method to incompressible Navier–Stokes equations, *J. Comput. Phys.* **59**, 308 (1985).
8. D. Kim and H. Choi, A second-order time-accurate finite volume method for unsteady incompressible flow on hybrid unstructured grids, *J. Comput. Phys.* **162**, 411 (2000).
9. R. Lohner, Mesh adaptation in fluid mechanics, *Eng. Fracture Mech.* **50**(5/6), 819 (1995).
10. M. J. Berger, M. J. Aftosmis, and G. Adomavicius, Parallel multigrid on Cartesian meshes with complex geometry, in *Proceedings of the Parallel CFD Conference 2000, Trodheim, Norway*.
11. J. H. Ferziger and M. Peric, *Computational Methods for Fluid Dynamics* (Springer-Verlag, Berlin/New York, 1997), p. 226.
12. V. Seidl, S. Muzaferija, and M. Peric, Parallel DNS with local grid refinement. *Flow Turbl. Combust.* **59**(4), 379 (1998).
13. F. H. Harlow and J. E. Welch, Numerical calculation of time-dependent viscous incompressible flow of fluids with free surfaces, *Phys. Fluids* **8**(12), 251 (1965).
14. C. M. Rhie and W. L. Chow, Numerical study of the turbulent flow past an airfoil with trailing edge separation, *AIAA J.* **21**, 1525 (1983).

15. P. J. Zwart, G. D. Raithby, and M. J. Raw, An integrated space–time finite volume method for moving boundary problems, *Numer. Heat Transfer B* **34**, 257 (1998).
16. Y. Zang, R. L. Street, and J. R. Koseff, A non-staggered, fractional step method for time-dependent incompressible Navier–Stokes equations in curvilinear coordinates, *J. Comput. Phys.* **114**, 18 (1994).
17. B. R. Hutchinson and G. D. Raithby, A multigrid method based on additive correction strategy, *Numer. Heat Transfer* **9**, 511 (1986).
18. M. Raw, Robustness of coupled algebraic multigrid for the Navier–Stokes equations, in *34th Aerospace Sciences Meeting and Exhibit, January 15–18, 1996, Reno, NV* (AIAA 96-0297).
19. R. B. Simpson, Anisotropic mesh transformations and optimal error control, *Appl. Numer. Math.* **14**, 183 (1994).
20. R. L. Rardin, *Optimization in Operations Research* (Prentice Hall, New York, 1998).
21. U. Ghia, K. N. Ghia, and C. T. Shin, High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method, *J. Comput. Phys.* **48**, 387 (1984).
22. C. S. Peskin, Flow patterns around heart valves: A numerical method, *J. Comput. Phys.* **10**, 252 (1972).
23. C. S. Peskin, Numerical analysis of blood flow in the heart, *J. Comput. Phys.* **25**, 220 (1977).
24. A. M. Roma, C. S. Peskin, and M. J. Berger, An adaptive version of the immersed boundary method, *J. Comput. Phys.* **153**, 509 (1999).
25. M.-C. Lai and C. S. Peskin, An immersed boundary method with formal second-order accuracy and reduced numerical viscosity, *J. Comput. Phys.* **160**, 705 (2000).
26. E. A. Fadlum, R. Verzicco, P. Orlandi, and J. Mohd-Yusof, Combined immersed-boundary finite difference methods for three-dimensional complex flow simulations, *J. Comput. Phys.* **161**, 35 (2000).
27. J. Mohd-Yusof, *Combined Immersed Boundaries/B-Splines Methods for Simulations of Flows in Complex Geometries*, CTR Annual Research Briefs (NASA Ames/Stanford University, Stanford, 1997).
28. C. H. K. Williamson, Oblique and parallel modes of vortex shedding in the wake of a circular cylinder at low Reynolds numbers, *J. Fluid Mech.* **206**, 579 (1989).