Towards large-eddy simulation in complex geometries

By K. Mahesh, G. R. Ruetsch AND P. Moin

1. Motivation and objectives

In this study we develop a numerical tool capable of performing large-eddy simulations in realistic engineering configurations. Such geometries are generally discretized using block-structured or unstructured grids; we have opted for unstructured grids because of the relative ease with which they are generated. The flow solver uses the finite difference approach and is motivated by the success enjoyed by the staggered-grid Harlow-Welch algorithm (1965) for large-eddy simulations on structured grids. This report describes our progress in developing and implementing the algorithm on a parallel platform.

2. Accomplishments

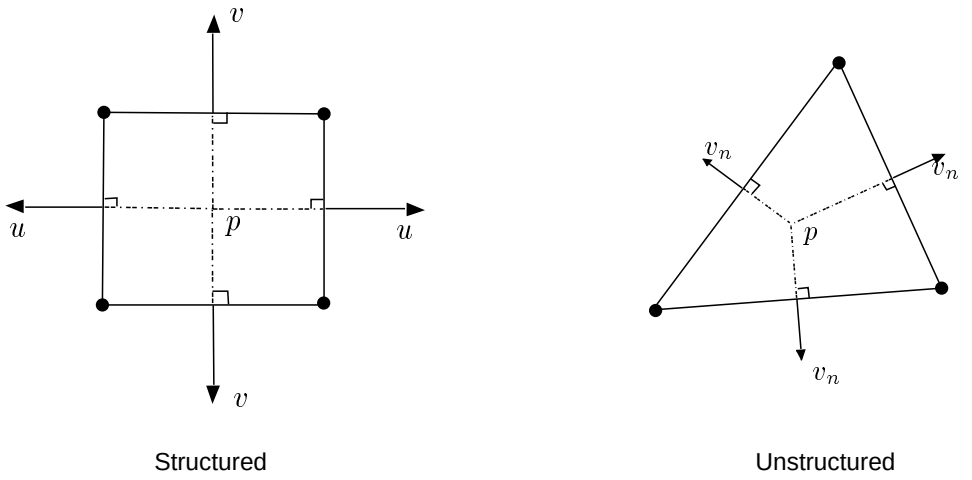
The Harlow-Welch (1965) staggered algorithm for structured grids was extended to unstructured grids of triangles in two dimensions and tetrahedra in three dimensions. Related approaches have been proposed in the literature; e.g. Amit *et al.* (1981), who considered triangular grids, and Nicolaides (1989, 1993, 1995). However, this extension (referred to as the circumcenter formulation) requires that a unique circumcenter be defined for the computational elements, making extension to arbitrary elements difficult. A more general formulation was, therefore, derived and implemented on a parallel platform for hybrid grids composed of tetrahedra, hexahedra, wedges, and prisms. The infrastructure needed to generate a hybrid grid for arbitrary geometry, compute the grid connectivity required, partition the grid, and specify arbitrary boundary conditions external to the flow solver was developed. The ordering of grid elements in unstructured grids can significantly influence single and multiple processor performance. To account for the possibility of dealing with unordered grids, an ordering algorithm that takes both single and multiple processor performance into consideration was developed.

Sections 2.1 and 2.2 describe the circumcenter and general formulations respectively. An overview of the solver is provided in section 2.3, which also discusses the reordering algorithm.

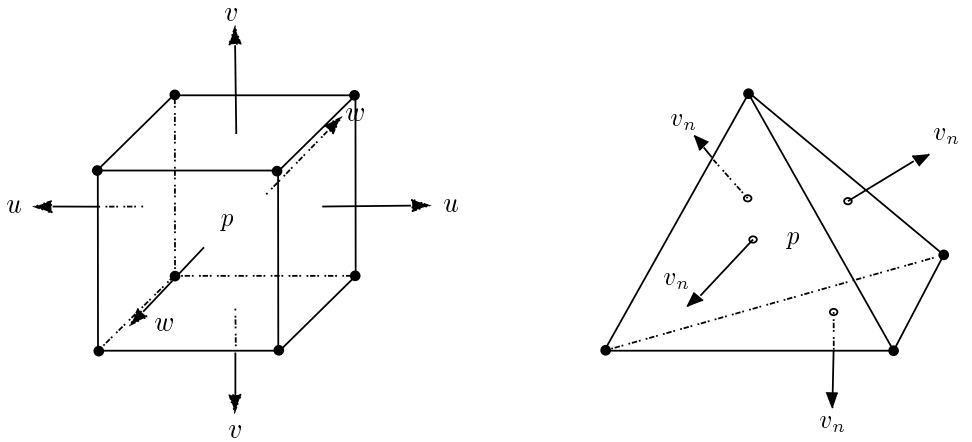
2.1 Algorithm description

The incompressible Navier-Stokes equations are given by

$$\nabla \cdot \vec{u} = 0,$$
$$\frac{\partial \vec{u}}{\partial t} + \vec{\omega} \times \vec{u} + \nabla \left(\frac{\vec{u} \cdot \vec{u}}{2} \right) = -\frac{1}{\rho} \nabla p + \nu \nabla^2 \vec{u}.$$



(a)



(b)

FIGURE 1. The staggered positioning of variables on unstructured and structured grids. (*a*) 2D. (*b*) 3D.

Figure 1 shows the staggered positioning of the velocity and pressure in the Harlow-Welch (1965) algorithm. Pressure is stored at the center of the Cartesian elements, while one component of the velocity is stored on each of the edges (in 2D) and faces (in 3D). This positioning of variables is easily extended to an unstructured grid. Note that the unstructured elements are assumed to be triangles in 2D and tetrahedra in 3D. The pressure is now stored at the circumcenter of the elements, while on each edge (in 2D) or face (in 3D), the velocity component normal to the edge (in 2D) or face (in 3D) is stored (Fig. 1).

The classical structured-grid algorithm may be interpreted in an unstructured manner as follows. On each edge (2D) or face (3D), we solve for the edge-normal (2D) or face-normal (3D) velocity. Tangential velocity components when needed are obtained by interpolating the normal velocities from the surrounding faces.

	Unstructured	Ghia *et al.*
Nodes	10576	66049
Elements	30925	65536
$\Delta_{\min}$	0.005	0.004
$\Delta_{\max}$	0.1	0.004

TABLE 1. The grid used in the unstructured computations ($Re = 5000$) is contrasted with that used by Ghia *et al.* (1982).

Defining pressure at the cell centers allows for a straightforward computation of pressure gradients at the faces. Also, solving for the face-normal velocities allows the discrete divergence at the cell-centers to be computed using the divergence theorem, which in turn allows the pressure in a pressure-projection approach to be consistent with the discrete continuity equation.

This interpretation forms the basis of the unstructured algorithm. If $\vec{n}$ denotes the normal to an edge (2D) or face (3D), the normal velocity component $v_n = \vec{u} \cdot \vec{n}$ satisfies

$$\frac{\partial v_n}{\partial t} - (\vec{u} \times \vec{\omega}) \cdot \vec{n} + \frac{\partial}{\partial n}\left(\frac{\vec{u} \cdot \vec{u}}{2}\right) = -\frac{1}{\rho}\frac{\partial p}{\partial n} + \nu\left(\nabla^2 \vec{u}\right) \cdot \vec{n}.$$

Note that the convection term is written in rotational form. This allows the circulation theorem to be imposed as a constraint on the algorithm. The vorticity at the nodes (2D) and edges (3D) is obtained from the normal velocities using the circulation theorem. This yields both the convection term as well as the viscous term since $\nabla^2 \vec{u} = -\nabla \times \vec{\omega}$ for incompressible flow. The velocity tangential to the edges (2D) or faces (3D) is obtained from the normal velocities on the neighboring edges (2D) or faces (3D). The tangential velocities on structured grids are obtained by an explicit interpolation; the unstructured formulation yields a system of equations for the tangential velocities. The pressure is obtained using a fractional-step procedure, which ensures that the discrete divergence is identically zero. Time-advancement is explicit and uses the second-order Adams-Bashforth method.

2.2 Evaluation of two-dimensional algorithm

2.2.1 Driven cavity flow

The two-dimensional circumcenter formulation is evaluated in this section for the flow in a driven cavity. The resulting flow comprises of a primary vortex that fills the cavity and secondary and tertiary vortices in the boundary layers near the corners.

The results are validated by comparing to benchmark computations by Ghia *et al.* (1982) using a structured grid. The quantities examined include velocity profiles, the vorticity at the center of the primary vortex, and visualization of the

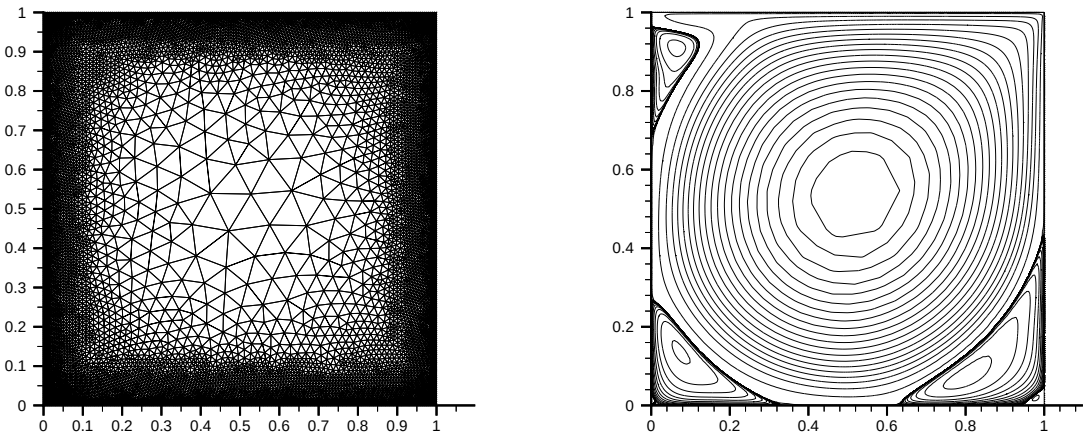


FIGURE 2. Grid (a) and streamfunction contours (b) for the flow in a driven cavity when $Re = 5000$.

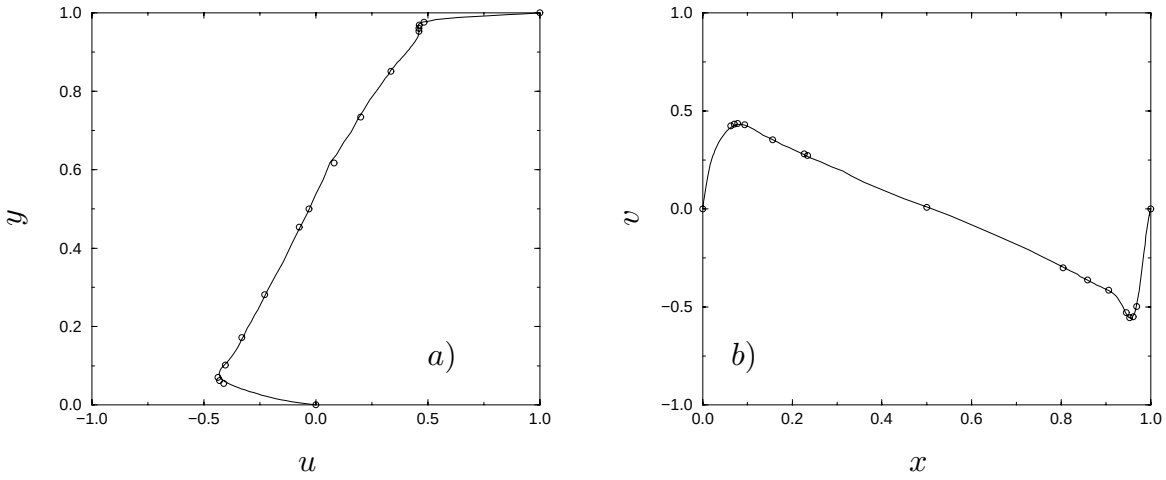


FIGURE 3. Unstructured results (solid lines) are compared to results from Ghia *et al.* (symbols). The Reynolds number is 5000. (a) Vertical profile of streamwise velocity at $x = 0.5$. (b) Horizontal profile of vertical velocity at $y = 0.5$.

streamfunction and velocity vectors. Computations were performed for Reynolds numbers (Uh/ν) varying from 400 to 5000; results for $Re = 5000$ are presented here.

Figure 2a shows the unstructured grid used in the computations. Note the coarseness of the grid in the center of the cavity. Also shown (Fig. 2b) are contours of the streamfunction. The secondary and tertiary vortices are seen to be accurately computed (see Fig. 3 in Ghia *et al.* 1982). Figures 3a and 3b provide a quantitative comparison of our results to those of Ghia *et al.* (1982). The unstructured computations are seen to yield the same quality of solution while using fewer points (see table 1).

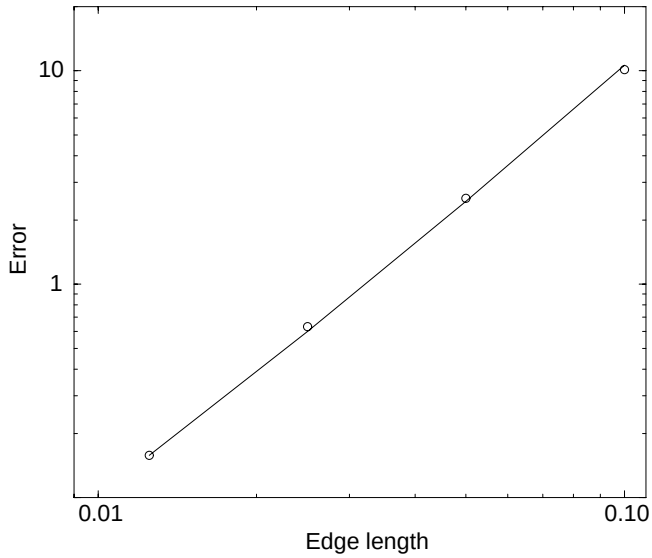


FIGURE 4. The percentage error in kinetic energy as a function of resolution. The solid line corresponds to a second-order accurate scheme while the symbols are from the computation.

2.2.2 Convergence study

A grid convergence study was performed to verify that the spatial discretization is indeed second-order accurate for uniform grids. The flow described by,

$$u = -\cos 2\pi x \, \sin 2\pi y \, \mathrm{e}^{-8\pi^2 t},$$
$$v = \sin 2\pi x \, \cos 2\pi y \, \mathrm{e}^{-8\pi^2 t},$$
$$p = -\frac{1}{4}\left(\cos 4\pi x + \cos 4\pi y\right) \mathrm{e}^{-16\pi^2 t}.$$

was used for this purpose. The solution was discretized on a square domain, using a grid of nearly equilateral triangles. The edge lengths were progressively refined from 0.1 to 0.0125 in factors of 2. A fixed CFL number of 1 was used to run the flow out to a dimensional time of 0.025. This corresponds to a non-dimensional time ($8\pi^2 t$) of 2. The kinetic energy decays by a factor of approximately 50 over this time. The fractional error, $100(q^2_{\mathrm{num}}/q^2_{\mathrm{an.}} - 1)$ at the end of the computation is shown in Fig. 4. The error in kinetic energy is seen to drop by a factor of 4 every time the mesh is refined, confirming that the spatial discretization is indeed second order.

2.3 Generalization of algorithm

The algorithm derived thus far is restrictive in that the pressure (and scalars if any) is stored at the circumcenter of the elements. This restricts the grid to

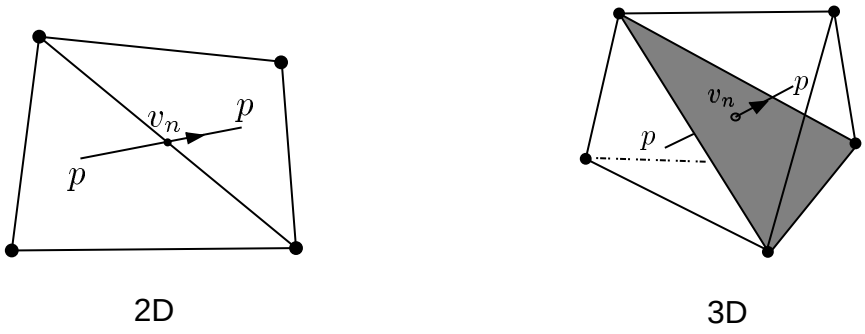


FIGURE 5. Schematic of the positioning of variables in the general formulation.

elements for which circumcenters can be defined and, furthermore, to circumcenters who lie within their elements. Not all elements have this property. For example, hexahedral elements do not possess a circumcenter. The circumcenters of elements that are isosceles right triangles lie on the hypotenuse and not inside. This issue is particularly important in three dimensions, where it was found that for even very simple geometries three-dimensional tetrahedral elements generated by grid generators do not necessarily have their circumcenters inside them. Furthermore, hexahedral elements were found preferable over tetrahedral elements. For the same nodal distribution, a hexahedral discretization requires less time to generate and has significantly fewer volumes than a corresponding tetrahedral discretization.

As a result, a more general formulation was derived. The pressure may now be located anywhere inside the element, allowing general grids (and grid generators) to be used. Two different versions were derived. Figure 5 shows the arrangement of variables in the first version. Given a location for pressure, the velocity at an edge (2D) or face (3D) is stored at the location where the line joining the pressure location of the element and its neighboring elements intersects the edge or face. Also, the velocity component stored is the component along the direction of the joining line. In three dimensions, this generalization loses some of the elegance of the circumcenter formulation. The face velocities no longer determine the edge-component vorticities. As a result, the face velocities are interpolated to the nodes and used to compute the face-normal vorticity prior to computing the convection and diffusion terms. In addition, the face velocity that is solved for does not determine the velocity-divergence for the elements. The pressure-projection step, therefore, involves the tangential velocities as well. The generalized algorithm was validated in two and three dimensions. Both steady and unsteady calculations were performed. In all cases the pressure was stored at the centroid. Results were obtained on grids for which the circumcenter formulation could not be used.

A second version of the general algorithm is currently being tested. In this version, pressure is stored at the centroids of the elements, and the face-normal velocity is stored at the centroids of the faces. A couple of reasons motivate this approach — the projection step is now independent of the tangential velocities, and storing velocities at the centroids was found to yield more accurate estimates of the fluxes across the face. Preliminary results are encouraging.

2.4 Solver details

The algorithm is implemented on parallel platforms using the Single Program Multiple Data (SPMD) paradigm and the MPI library for message passing. This approach requires that the following issues be addressed.

2.4.1 Grid generation and domain decomposition

Grid generation and domain decomposition are performed using third-party software. The code contains the infrastructure required for handling hybrid grids that are composed of tetrahdera, hexahedra, prisms, and wedges. Once generated, the required grid connectivity is computed, and the grid is decomposed into partitions that reside on individual processors. The quality of the decomposition is determined in terms of the degree of load balancing, where each partition should contain the same number of elements, and the communication bandwidth. The package, 'METIS' (Karypis & Kumar, 1997) was found to produce good quality partitions for unstructured grids both in terms of load balancing and bandwidth.

2.4.2 Grid ordering

The order in which grid entities are represented in memory has a significant effect on both serial and parallel performance. Parallel performance is generally enhanced by ordering the control volumes into three main categories: interior, boundary, and ghost. By using such a *macroscopic* ordering scheme, it is possible to overlap computation and communication: while interprocessor data transfer occurs from boundary to ghost volumes, computations requiring data from native elements (i.e. interior and boundary elements) can be performed on interior volumes. In addition to macroscopic reordering, one also has considerable flexibility in reordering the internal elements in order to enhance cache reuse and, hence, serial performance. Such *microscopic* reordering is commonly considered independent of the *macroscopic* reordering. However, one can achieve better orderings if the dependencies between the macroscopic and microscopic orderings are considered and exploited.

Figure 6 shows how different ordering strategies affect data locality. The Laplacian matrix is used for illustration purposes. As an example of what can occur when macroscopic and microscopic reorderings are considered independently, consider the unstructured Laplacian matrix that is obtained when the following ordering strategy (global reordering) is used: microscopic level-set reordering on the global grid followed by grid partitioning and then macroscopic reordering. Note that level-set reordering is a technique where the order of elements is determined as they come into contact with a level set (propagating front) that sweeps through the domain. Comparison to Fig. 6a (which is not microscopically reordered) shows significant improvement in data locality. (Ideally, one would want a banded diagonal system, as in the case of structured grids.) However, the interaction between the interior and boundary elements is poor, indicated by the near vertical and horizontal lines in the boundary regions. This poor data locality between interior and boundary nodes is a direct consequence of treating the microscopic and macroscopic ordering independently.

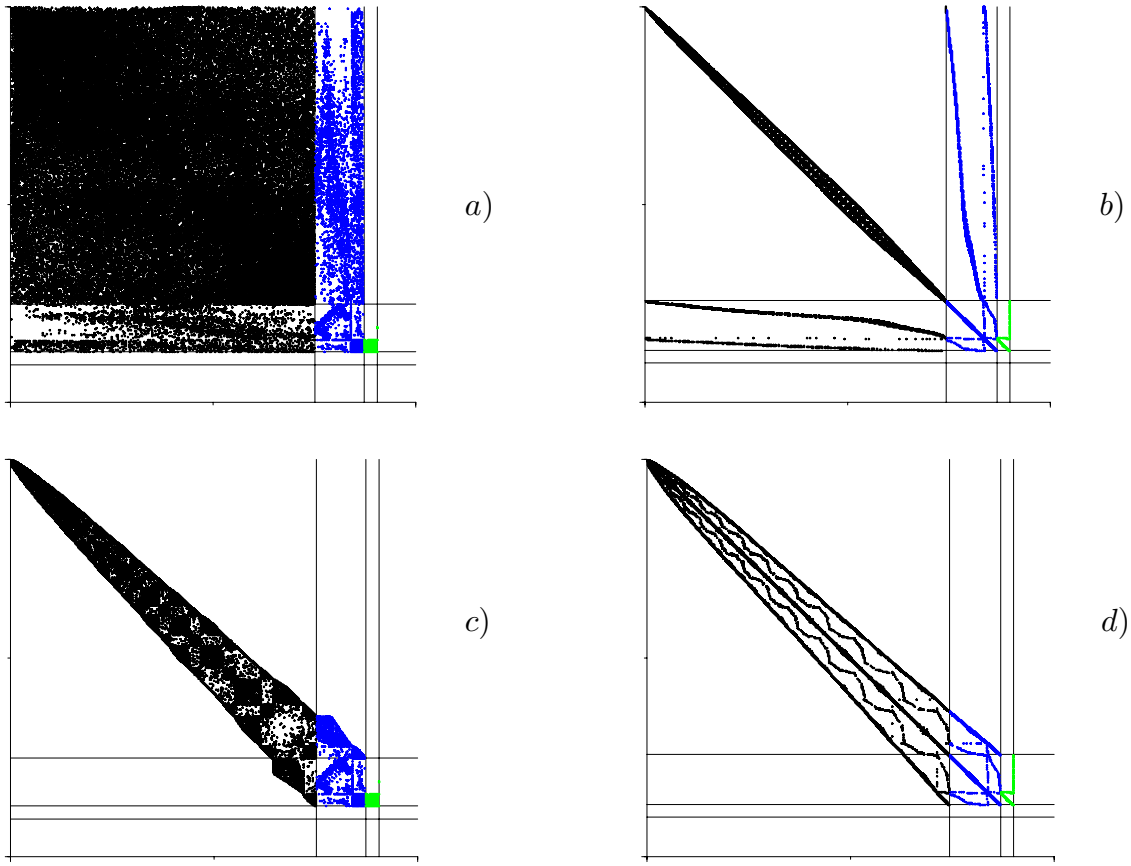


FIGURE 6. Unstructured Laplacian matrices for various reordering schemes: unordered (a), global preordered (b), reverse level-set reordered (c), double reordered (d). The vertical and horizontal lines separate the interior, boundary, and ghost zones (plus a fourth zone that is not addressed here).

The results of a simple method (referred to as reverse level-set reordering) to improve the interior-boundary data locality is demonstrated in Fig. 6c. Here, the grid is first partitioned, then macroscopically reordered, and finally microscopically reordered using a level-set method *initiated at the boundary*. Although this method does solve the interior-boundary data locality issue, the unstructured Laplacian has the undesirable feature that the maximum bandwidth is fairly large (equal to the number of boundary elements on the partition).

While there is not much one can do to improve the bandwidth required for a reverse level-set technique, the fill within the bandwidth can be greatly improved. The fill in the reverse level-set reordering method is large because it inherits the poor ordering of the global grid through the boundary elements. The poor ordering of the boundary elements can be improved by applying an additional microscopic reordering to the global grid. Thus, we have a procedure, called double reordering, which consists of a level-set reordering before the partitioning and a reverse-level set reordering after the macroscopic reordering. The first microscopic reordering seeds the partition boundary with a good ordering, and the second achieves a good

interior-boundary element data locality. The results of double reordering are shown in Fig. 6d, where, although there is no improvement in bandwidth over the reverse level-set technique, the fill is significantly improved.

For the relatively small grids tested thus far, all of the reordering schemes show comparable performance. However, as larger grids are encountered and when scalar fields are introduced (in interlaced arrays), we expect to see a noticeable improvement in performance from the doubly reordered grids.

Acknowledgments

We are grateful to Professor Blair Perot for useful discussions during the course of this project.

REFERENCES

Amit, R., Hall, C. A. & Porshing, T. A. 1981 The dual variable method for solving fluid flow difference equations on Delaunay triangulations. *J. Comp. Phys.* **40,** 183.

Ghia, U., Ghia, K. N. & Shin, C. T. 1982 High-Re solutions for incompressible flow using the Navier-Stokes equations and a multigrid method. *J. Comp. Phys.* **48,** 387.

Harlow, F. H. & Welch, J. E. 1965 Numerical calculation of time-dependent viscous incompressible flow of fluid with free surface. *Phys. Fluids.* **8,** 2182.

Karypis, G. & Kumar, V. 1997 Metis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices, version 3.0. *Technical report.* Dept. of Computer Science, University of Minnesota, Oct 1997.

Nicolaides, R. A. 1989 Direct discretization of planar div-curl problems. *ICASE Report 89-76.*

Nicolaides, R. A. 1993 The covolume approach to computing incompressible flow. *Incompressible Computational Fluid Dynamics.* Cambridge University Press, Gunzburger, M. D. & Nicolaides, R. A. eds., 295.

Nicolaides, R. A. & Wu, X. 1995 Covolume solutions of three-dimensional div-curl equations. *ICASE Report 95-4.*