INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN FLUIDS
Int. J. Numer. Meth. Fluids 2001; **36**: 519–538 (DOI: 10.1002/fld.143)

A multiblock/multilevel mesh refinement procedure for CFD computations

Rune Teigland*,[1] and Inge K. Eliassen[2]

Department of Mathematics, University of Bergen, Bergen, Norway

SUMMARY

A multiblock/multilevel algorithm with local refinement for general two- and three-dimensional fluid flow is presented. The patched-based local refinement procedure is presented in detail and algorithmic implementations are also presented. The multiblock implementation is essentially block-unstructured, i.e. each block having its own local curvilinear co-ordinate system. Refined grid patches can be put anywhere in the computational domain and can extend across block boundaries. To simplify the implementation, while still maintaining sufficient generality, the refinement is restricted to a refinement of the grid successively halving the grid size within a selected patch. The multiblock approach is implemented within the framework of the well-known SIMPLE solution strategy. Computational experiments showing the effect of using the multilevel solution procedure are presented for a sample elliptic problem and a few benchmark problems of computational fluid dynamics (CFD). Copyright © 2001 John Wiley & Sons, Ltd.

KEY WORDS: composite grids; local refinement; multilevel methods

1. INTRODUCTION

In the simulation of complex industrial flows the main challenge consists in modelling and computing phenomena on a wide range of scales, from a scale that is very small, as compared with the global computational domain, up to scales that are of the size of the computational domain. Multiblock methods are often employed to compute flows in complex geometries. The multiblock approach (in three-dimensional form) is to segment the physical region into contiguous subregions, each bounded by six curved sides and each of which transforms to a cubic block in the computational region. Each grid block is assumed to be topologically cubic (three-dimensional) and has its own curvilinear co-ordinate system.

* Correspondence to: Norsk Hydro ASA, Research Centre, Sandsliveien 90, N-5020 Bergen, Norway.
[1] E-mail: resrt@mi.uib.no
[2] E-mail: Inge.Eliassen@mi.uib.no

Received August 1999
Revised November 1999

Many problems in the field of partial differential equations (PDEs) exhibit solution behaviours that require more resolution in one area of the domain than in others. There are often also large portions of the domain where high levels of refinement are not needed. To achieve an effective numerical simulation of the physical processes, it is important to use a flexible discretization adapted to the phenomena one wants to describe. The challenge is in the modelling of phenomena that are present at a scale much smaller than the global computational domain. Local grid refinement is a natural strategy. With a global and relatively coarse grid that is supposed to catch the main features of the phenomena studied, one adds extra degrees of freedom in regions where they are needed. In this way one hopes to distribute the computational work in accordance with the properties of the physical model, thus achieving optimal efficiency. The basic idea is to start off from a standard coarse base grid and then to embed multiple layers of grid refinement by successively dividing the control volumes in the designated region. Grid refinement can be static or dynamic.

2. GOVERNING EQUATIONS AND DISCRETIZATION

In this section a general framework for fluid flow in complex three-dimensional geometries is presented. The flow of Newtonian fluids is governed by the Navier–Stokes equations, which express conservation of mass and momentum. In the case of steady incompressible, laminar and isothermal flow these can be written

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\nabla \cdot (\rho \mathbf{u} u_l) = -\frac{\partial p}{\partial x_l} + \mu \nabla^2 u_l + f_l, \quad l = 1, 2, 3 \tag{2}$$

where ρ is the density, $\mathbf{u} = (u_1, u_2, u_3) = (u, v, w)$ represents the velocity field, p is the pressure and μ is the dynamic viscosity of the fluid. External forces and other source terms are denoted by f.

In the case of turbulent flows, the equations are augmented by equations using a standard k–ε turbulence model. The conservation equations may be written in general form as follows:

$$\frac{\partial(\rho\phi)}{\partial t} + \frac{\partial(\rho u_j \phi)}{\partial x_j} = \frac{\partial}{\partial x_j}\left(\Gamma_\phi \frac{\partial \phi}{\partial x_j}\right) + S_\phi \tag{3}$$

where ϕ represents different conserved quantities, such as momentum, continuity, turbulent kinetic energy, etc. In the steady state case this is a prototype of a scalar advection–diffusion equation. These equations are discretized on a non-orthogonal colocated grid arrangement. If a flux vector $\vec{J}_j$ containing convection and diffusion is defined as

$$\vec{J}_j = \rho u_j \phi - \Gamma_\phi \frac{\partial \phi}{\partial x_j} \tag{4}$$

Equation (3) can be written as

$$\frac{\partial(\rho\phi)}{\partial t}+\frac{\partial \vec{J}_j}{\partial x_j}=S_\phi \tag{5}$$

We use a finite volume discretization scheme where the equations for the conserved quantities are integrated over a general non-orthogonal control volume. Multiblock grids are handled using a local co-ordinate system. Thus, the discretization of the equations proceeds block by block, exactly as in the single block case. Implicit time discretization (backward Euler) for the transient term yields

$$\int_{\delta V}\frac{\partial(\rho\phi)}{\partial t}=\frac{(\rho_p\phi_p-\rho_p^0\phi_p^0)}{\Delta t}\,\delta V_p \tag{6}$$

where superscript '0' denotes values from the previous time step. The general conservation equation for ϕ, Equation (3), is integrated over a three-dimensional control volume δV_p in physical space such that after employing the Gauss divergence theorem, one gets

$$\frac{(\rho_p\phi_p-\rho_p^0\phi_p^0)}{\Delta t}\,\delta V_p+\sum_{nn}\vec{J}\cdot\vec{A}|_{nn}=S_p \tag{7}$$

where nn denotes the cell face index, p is the cell-centre index, $\vec{A}$ is the area vector and S_p is the total source in the control volume. Because the grid is non-orthogonal, the derivatives that occur in the viscous and pressure terms must be evaluated in the transformed curvilinear co-ordinate system (ξ_1, ξ_2, ξ_3). If we use the chain rule for differentiation, we get

$$\frac{\partial\phi}{\partial x_j}=\sum_l\frac{\partial\xi_l}{\partial x_j}\frac{\partial\phi}{\partial\xi_l}=\sum_l e^l\frac{\partial\phi}{\partial\xi_l} \tag{8}$$

where e^l are the contravariant basis vectors of the curvilinear co-ordinates. Details of the derivation can be found in References [1,2]. In order to alleviate checkerboard oscillations in pressure (due to the use of a colocated grid arrangement) we use a pressure weighted interpolation of the cell-face velocities in the discretized continuity equation. The idea goes back to Rhie and Chow [3].

The coupled momentum and continuity equations are solved in a sequential manner using the SIMPLE method of Patankar and Spalding [4]. Notice that the arising linear system for each component of the velocity is non-symmetric. The SIMPLE solution method belongs to the well-known class of pressure correction methods popular in the primitive variable computation of complex incompressible flows [5]. The methods use a predictor–corrector approach to the solution of the Navier–Stokes equations. The SIMPLE method is widely used in general purpose computational fluid dynamics (CFD) codes. It is a robust methodology applicable for complex, turbulent recirculating flows and it is potentially applicable to all regimes from incompressible laminar flows to supersonic flows. The basic SIMPLE solution strategy consists of iteratively solving the momentum and the pressure correction equations.

The solution of scalar equations, such as equations for the turbulence production rate and turbulence dissipation rate, are then solved using essentially the same basic form of Equation (3). The complete main loop in each time step is given by Algorithm 2.1.

Algorithm 2.1 (Main loop)

```
Update primary and derived fields, time, boundary conditions, etc.
Iteration loop:
  Solve u-velocities
  Solve v-velocities
  Solve w-velocities
  Calculate advective fluxes by Rhie and Chow interpolation
  Solve pressure correction
    Update pressure, velocity, advective fluxes and density
Repeat (if necessary)
Solve scalar fields
```

When only the steady state solution is of interest, the time step Δt is used as a parameter through which the convergence rate may be optimized. The viscous flux, see Equation (7), is split in a primary flux that contains the orthogonal terms (relating to grid) which are treated implicitly while the other terms are treated explicitly (i.e. lumped into the source terms). Thus, the arising system of linear equations takes the form

$$a_p \phi_p = \sum_{\text{nb}} a_{\text{nb}} \phi_{\text{nb}} + S_p \tag{9}$$

where each point p is coupled to its six neighbours (nb) and S_p denotes the discretized source term. The source term is split such that the resulting matrix associated with the linear system (9) is an M matrix with constant bandwidth. The M matrix property is important for the existence of the incomplete factorization method used as pre-conditioner in connection with the Krylov subspace methods [6].

3. DESCRIPTION OF A MULTIBLOCK/MULTILEVEL ALGORITHM

This section discusses some implementational and computational issues regarding Algorithm 2.1. A description is given of how the grid is made out of patches, what information is associated with each patch and how patches communicate. The code is implemented in the FORTRAN 90 programming language and makes extensive use of FORTRAN's new features of pointers and dynamic memory allocation.

The composite grid is made up by several blocks that are rectangular and in general curvilinear. Each patch has associated with it an element of TYPE(patch). This type contains field variables, boundary conditions and information on how the patches relate to each other hierarchically and how they overlap. The field variables include the unknowns, derived and interpolated fields, grid specifications and coupling coefficients for the linearized equations.

The arrays holding these fields are extended with two layers of ghost cells in all directions, which are used for inter patch communications.

The patches are organized in a hierarchy with levels $1, \ldots, L$. The coarsest patches on level 1 make up the computational domain. They can be Cartesian or curvilinear by specification of the cell corner co-ordinates. It is assumed that these coarsest patches have cells matching exactly in overlapping areas. Patches on level 2 are constructed by halving the grid size on level 1 in some or all directions. Patches on higher levels are constructed similarly. Figure 1 shows a two-dimensional example with three levels. For each level there is a linked list of pointers containing all the sibling patches on that level. In addition, the list of siblings on level l has pointers to the list of parents on level $l-1$ and the list of children on level $l+1$. This is illustrated in Figure 2, where the squares represent patches.

Each patch must know with which other patches it overlaps and how. Since the patches are topologically rectangular, the overlap region between two patches is also a rectangle and can thus be specified by the lower left and upper right corners. A patch P overlapping with another patch Q needs to know the rectangular region where its interior and ghost cells are overlapping with the interior cells only of the other patch Q, as illustrated in Figure 3. This region must be specified in local co-ordinates of both P and Q together with interpolation coefficients. This information is stored in an element of TYPE(overlap). All the overlap elements describing how P overlaps with other patches are organized in linked lists. There is one list for each of the levels of parents, siblings and children.

Finally, a patch needs to know which boundary conditions to apply on its six faces. The different types of boundary conditions include (1) wall—a physical wall with the no slip

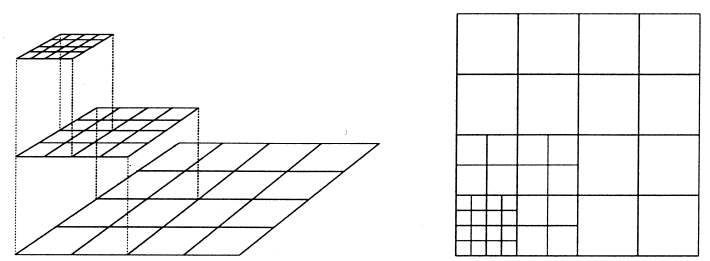


Figure 1. The composite grid has patches on several levels.

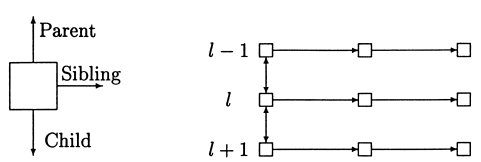


Figure 2. The patches are oganized in linked lists.

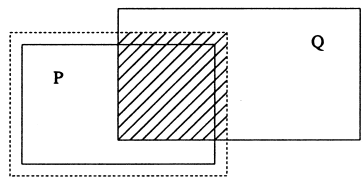


Figure 3. Overlap region between two patches as seen by P.

condition, (2) inlet—a fully specified flow into the domain, (3) outlet—flow out of the region, (4) symmetry, and (5) interior—the patch face is in the interior of the entire computational domain. Any one face of a patch may be governed by several of these conditions applied in different areas. A patch face is, therefore, divided into non-overlapping rectangles on each of which a single condition applies. These rectangles are organized in linked lists—one for each of the six faces.

Assembling the coupling coefficients and source term for the linearized momentum equations (2) is done with one sweep over all patches with very little communication between patches. When solving these equations more inter-patch communication is required. A patch solves an equation only on the area that is not covered by any children—elsewhere the solution is determined by finer patches. So when solving the linear system $A\phi = b$ a patch needs to get ϕ from parents and siblings to its ghost cells and from children to the interior and ghost cells they cover. When getting ϕ from parents and children, tri-linear interpolation, as discussed in Reference [7], is used. The interpolated value ϕ^P in a cell centre in a patch P is a weighted sum of ϕ^Q in eight surrounding cell centres in the overlapping patch Q

$$\phi^P = \sum_{i=1}^{8} N_i \phi_i^Q$$

The eight interpolation coefficients, N_i, for each cell of P in the overlap region are stored in the corresponding element of TYPE(overlap) and are determined once and for all prior to the start of the simulation. This way of patch communication is known as Dirichlet coupling from [8].

Two alternative methods to tri-linear interpolation from coarse to fine are discussed in Reference [7]. One of the possibilities is using the higher-order scheme of tri-quadratic interpolation. It is more accurate than tri-linear interpolation but is also more costly, both in terms of storage and computations, because of the wider stencil, which in three dimensions involves 27 coarse cells. The other possibility is referred to as compact interpolation. It consists of estimating a fine ghost cell value

$$\phi_{\text{ghost}} = \phi_{\text{interior}} + \nabla\phi_{\text{coarse}} \cdot \Delta\vec{r}$$

based on the gradient estimated on the coarse grid.

For solving $A\phi = b$ on the composite grid we use the method of BICGSTAB with line Gauss–Seidel iterations in all directions on each patch as preconditioning.

Conservation of mass is a key issue and to insure this care must be taken at the interface between coarse and fine patches. The fluxes along this interface are calculated by pressure weighted interpolation (see Reference [3]) on both the coarse and fine patches, as illustrated in Figure 4. These fluxes do not necessarily match because of discretization errors so that mass could be lost or gained. To insure mass conservation the flux through a coarse cell face is therefore corrected to equal the sum of the fluxes through the corresponding fine cell faces.

One complication arising because of local refinement is the treatment of coupling coefficients between the velocity correction and the pressure correction gradient on the coarse–fine interface. The coefficient C in

$$u' = -Cp'_x$$

is at a cell centre constructed from the linearized momentum equation coefficients at that cell centre. Linear interpolation determines C at cell faces. The problem for the fine patch at a coarse–fine interface is that there are no values for C in the ghost cells to use for the interpolation. A solution to this is extrapolating C from interior cells.

Another coarse–fine interface complication is the creation of artificial divergence due to the coarse grid flux correction. What happens is that first the velocities and fluxes are adjusted after solving the pressure correction equation. This ensures that the continuity equation is satisfied, i.e. for each cell $\Sigma F_i = 0$, where F_i is the flux across a cell face and the sum is taken over all six cell faces. Then the coarse grid flux correction is carried out. Altering the coarse grid fluxes on the coarse–fine interface creates a flux imbalance, $\Sigma F_i \neq 0$, for the coarse cells next to this interface. As a remedy it is possible to implement a different coupling between coarse and fine patches, which lets the coarse–fine interface flux on the fine patch dictate the flux on the coarse patch during the solution process of the pressure correction equation. This way there will be no need for coarse grid flux correction, so no artificial imbalance is created.

Using the flux as a condition on the boundary to another patch is known from Reference [8] as Neumann coupling. The reason for using this term is that a known flux implies zero flux correction, which again implies zero pressure correction gradient, i.e. homogeneous Neumann

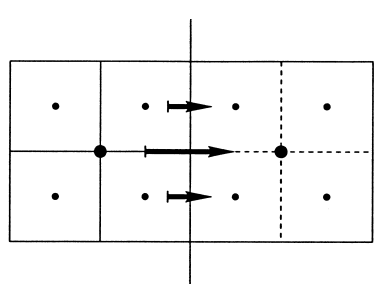


Figure 4. Fine and coarse fluxes at the same face must match.

condition. Combining Dirichlet (D) and Neumann (N) coupling gives several possibilities; DD, DN and NN. Figure 5 shows the stencils for the DN coupling, where the fine patch above the interface uses Dirichlet coupling and the coarse patch below uses Neumann coupling. Letting the coupling both ways be of Dirichlet type (DD) is easiest to implement, but there might be a problem with satisfying the continuity equation as discussed above. Neumann coupling both ways (NN) solves that problem, but introduces another. An extra coupling condition for p' across patch interfaces must be introduced since there are only one-sided zero gradient conditions at both patches. Dirichlet coupling one way and Neumann coupling the other (DN) allows a strong coupling of p' across patch interfaces at the same time as the coarse grid flux correction problem is solved.

A complication with the DN coupling is how to efficiently solve the pressure correction equation. The Neumann coupling alters the term S_p in Equation (9) for the coarse cells next to the coarse–fine interface. The equation thus changes dynamically as it is solved, and the solution method used for DD coupling may fail to converge. Alternatively, Algorithm 3.1 can be employed, but slower convergence may become a serious problem for large grids.

Algorithm 3.1 (Solving $Ap' = b$ using DN coupling)

```
Patch loop (fine to coarse):
  Interpolate solution to ghost cells from other patches
  Interpolate solution to entire patch from children
  Correct fluxes at coarse-fine boundary
  Adjust b for cells adjacent to the coarse-fine boundary
  Solve for p' where patch has no children
Repeat (if necessary)
```

The use of a locally refined grid naturally raises the question about how to get the best trade-off between accuracy and computational costs. In particular, one may ask where the fine grid patches should be placed and how many levels should be used. For relatively simple problems it may be known *a priori* where additional resolution is required. In these cases, static grid hierarchies may be designed with focus on these areas and with levels of refinement resulting in acceptable computational complexities. This is the approach taken in the experiments in the following section.

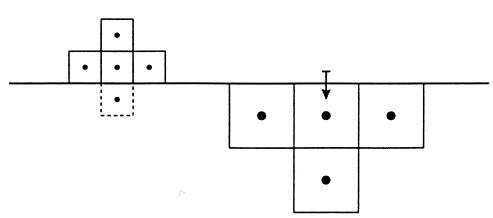


Figure 5. Stencils for Dirichlet and Neumann couplings.

More sophisticated techniques using adaptive refinements in space and time are possible [9–11]. The area to be refined may be determined automatically by using some kind of error estimation. Simple error indicators may be based on velocity gradients or curvature. Alternatively, estimates of the truncation error on coarser grid may be calculated [11] or some type of Richardson extrapolation may be used to estimate the discretization error [12]. Once the region requiring refinement has been identified, a compromise has to be made between using many patches to cover as little area as possible and using few patches covering a larger area in order to reduce the overhead associated with communication between patches.

4. NUMERICAL RESULTS

Three different test problems have been studied

- The Poisson problem—a scalar equation.
- Lid-driven cavity—a closed recirculating problem.
- Backward-facing step—a problem with open boundaries.

4.1. The Poisson problem

The Poisson equation examined here is $-\nabla^2\phi = S_\phi(x, y)$, with the source term $S_\phi(x, y) = -200\,\mathrm{e}^{-10(x+y)}$. The solution on the unit square with appropriate Dirichlet boundary conditions is $\phi(x, y) = \mathrm{e}^{-10(x+y)}$ and is plotted in Figure 6[3]. The Poisson equation is derived from the general conservation equation (3) by letting $u_j = 0$, $\Gamma_\phi = 1$ and considering the steady state solution. As the initial guess we use $\phi = 0$.

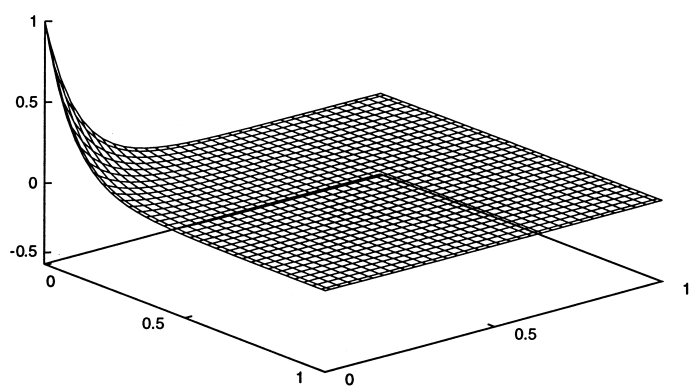
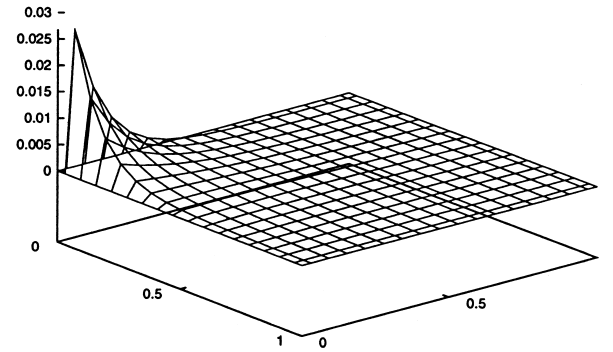
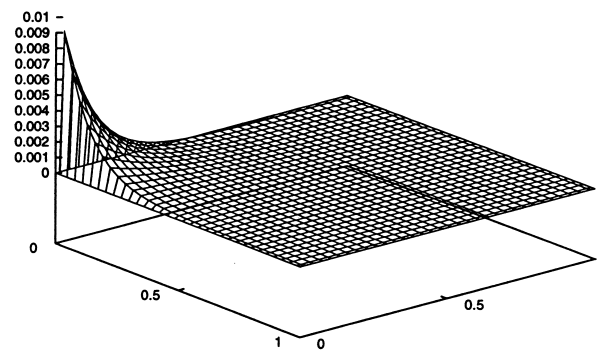


Figure 6. The solution of the Poisson problem on a 32×32 grid.

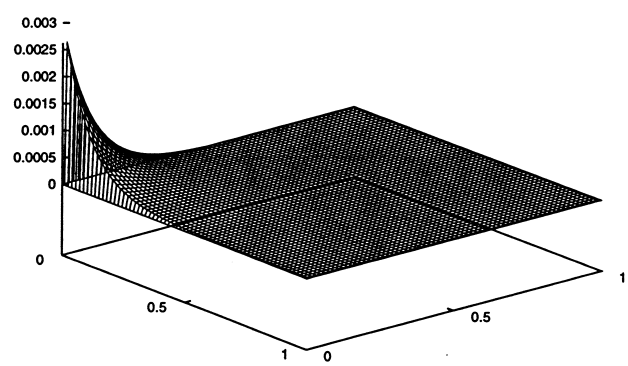
[3] In this and all subsequent plots of this kind, the values at cell centres are connected with lines. These lines do not correspond to cell faces.



(a) 16 × 16



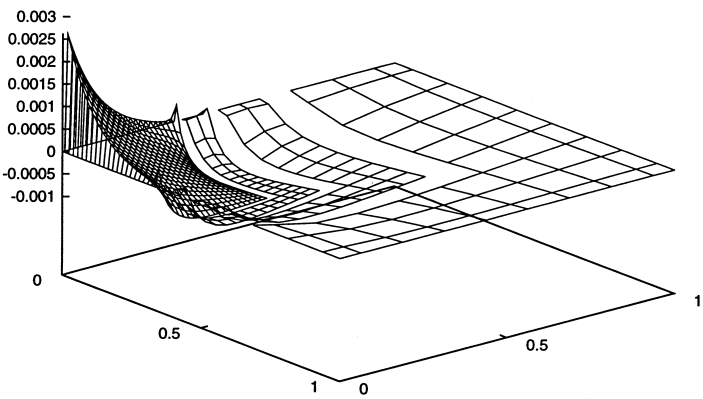
(b) 32 × 32



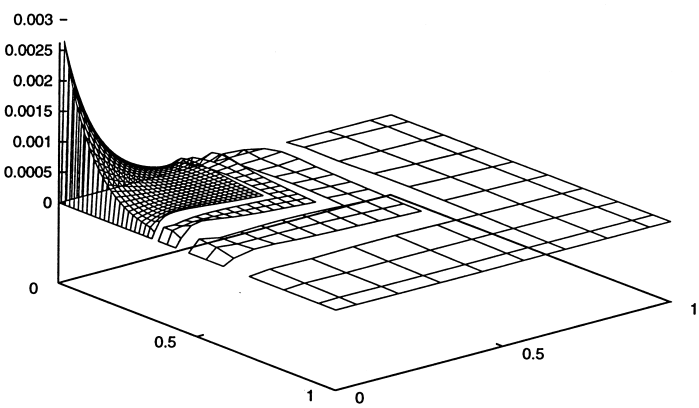
(c) 64 × 64

Figure 7. Discretization errors for uniform grids.

The discretization errors on uniform 16×16, 32×32 and 64×64 grids are plotted in Figure 7. We then compare with errors achieved using local refinement. Our coarsest grid is a uniform 8×8 grid and we have four grid levels, so the finest grid size corresponds to the uniform 64×64 grid in Figure 7(c). The refinements are placed near the origin, where the solution has steep gradients. Figure 8 shows the errors using bilinear and compact interpolation. We see that the error on the locally refined grid is comparable with the error on the uniform 64×64 grid even though the number of grid points is reduced by approximately 80 per cent. The corresponding reduction in computational time is about 65 per cent, reflecting the additional costs of maintaining the grid hierarchy.



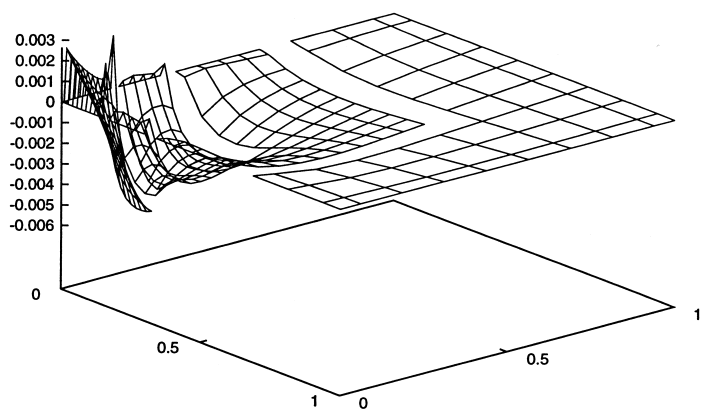
(a) Bilinear interpolation



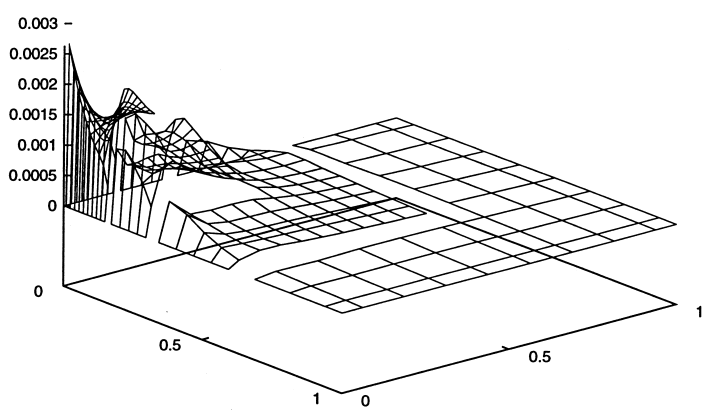
(b) Compact interpolation

Figure 8. Discretization errors for locally refined grids.

Getting good results is highly dependent on the choice of grid refinement. The refined regions should be as small as possible to save on computational efforts, but not so small that the accuracy deteriorates. If we choose the refined grids on levels 3 and 4 to be smaller, we get the results shown in Figure 9. The coarse–fine grid interfaces now have steeper solution gradients. This leads to larger errors, especially for the bi-linear interpolation method, as pointed out in Reference [7]. When comparing the two interpolation methods, the compact scheme seems more robust and accurate than the bi-linear scheme.



(a) Bilinear interpolation



(b) Compact interpolation

Figure 9. Discretization errors for locally refined grids, with smaller refined regions.

4.2. Lid-driven cavity

The lid-driven cavity is an extensively studied standard test case for flow simulation codes. The numerical solutions presented in Reference [13] are believed to be highly accurate and are often used as reference solutions for benchmark tests. The set-up for the lid-driven cavity in two dimensions is schematically depicted in Figure 10. The no-slip condition applies to all walls, and the upper wall has a constant velocity to the right. Viscosity forces create a clockwise circulation and, depending on the Reynolds number, higher-order eddies in the corners. The Reynolds number is in this case defined as $Re = UL/v$, where U and L are the speed and length of the moving wall respectively, and v is the kinematic viscosity. In our experiments we have chosen $Re = 1000$.

Figure 11 shows some test results on a uniform 64×64 grid. The figure shows the centreline velocities $u(0.5L, y)$ and $v(x, 0.5L)$. Various schemes for flux treatments are compared with the reference solution in Reference [13]. The first-order upwind scheme performs worst. The two other alternatives, $\kappa = 1/3$ and 'Quick', are higher-order schemes implemented as explicit corrections to the upwind scheme. They both lead to significant improvements. The $\kappa = 1/3$ scheme seems to give somewhat better results.

To test the local grid refinement procedure we have repeated a numerical experiment described in Reference [14]. One of the grids used here is shown in Figure 12. It consists of two grid blocks with a 2:1 cell size ratio along the interface. The top block is 64×32 and the bottom is 32×32. The simulation is also done with the same grid layout with twice as fine resolution. The centreline velocities, plotted in Figure 13, are in agreement with the results in Reference [14].

A second experiment with a skewed lid-driven cavity was carried out as a test for using non-orthogonal and stretched grids. The set-up was implemented as in Reference [15] for $Re = 1000$ and the angle between grid lines $\beta = 30°$. Several grid sizes and different amounts of grid stretching were tested. We also examined the possibility of placing locally refined grid

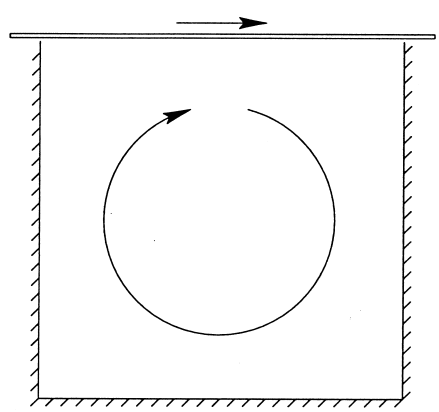


Figure 10. Lid-driven cavity.

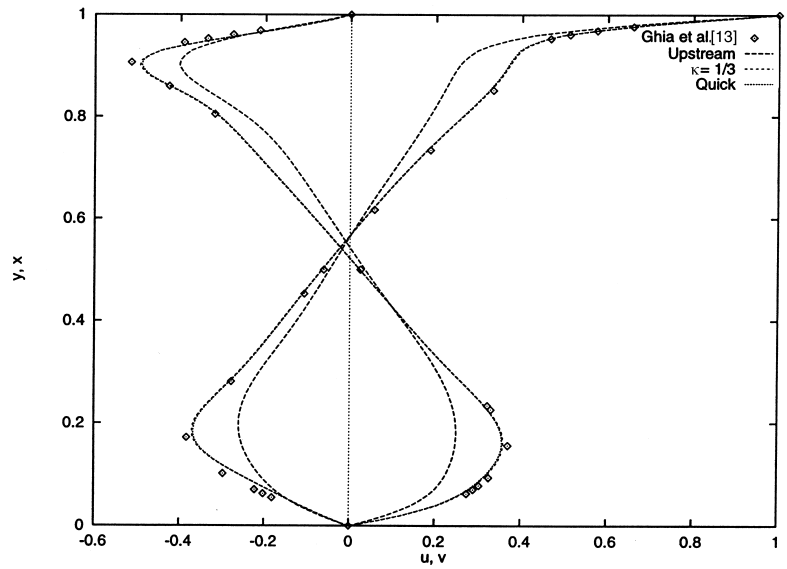


Figure 11. Centreline velocities for a 64×64 grid with various schemes for explicit flux correction.

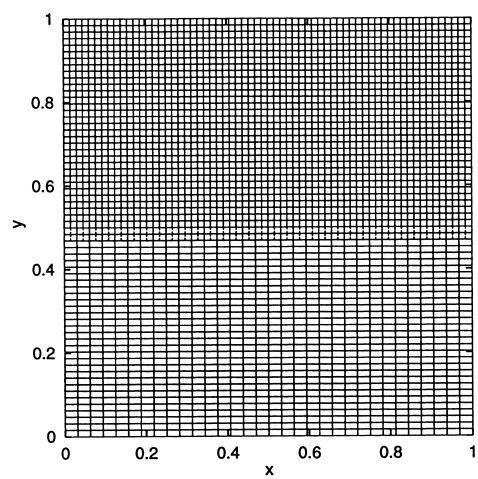


Figure 12. The grid layout from Reference [14].

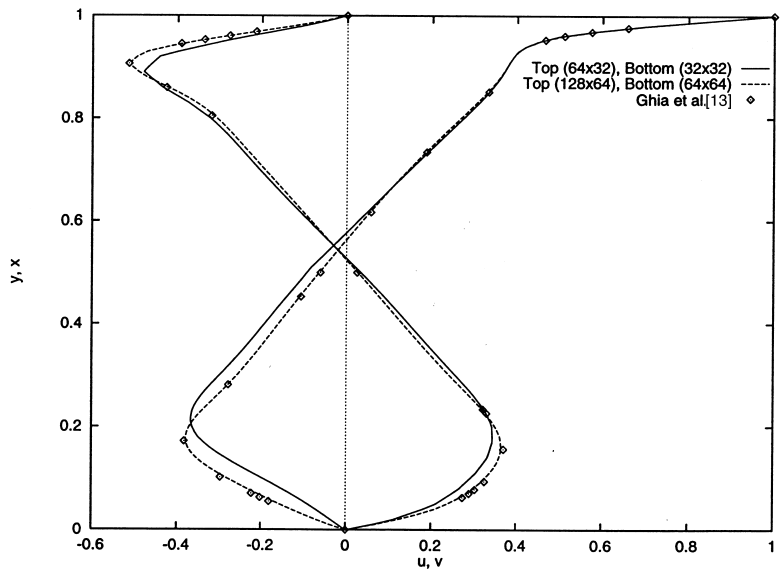


Figure 13. Centreline velocities with grid layout as in Figure 12.

patches along the moving lid and other regions, where enhanced spatial resolution could be expected to give more accurate results. However, this did not seem to lead to improved accuracy compared with a uniform grid solution of comparable computational complexity. A grid with carefully chosen stretching is capable of producing improved results compared with an equally sized regular uniform grid without any additional computational expenses, other than possibly the need for a time step reduction due to smaller grid cells. Figure 14 shows the stretching that produced best results for this test case. The plots of centreline velocities in Figures 15 and 16 demonstrate the improvements of stretched grids compared with regular

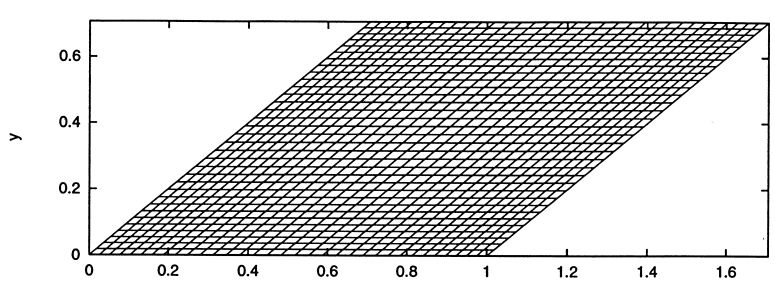


Figure 14. Stretched 33×33 grid for the skewed lid-driven cavity.

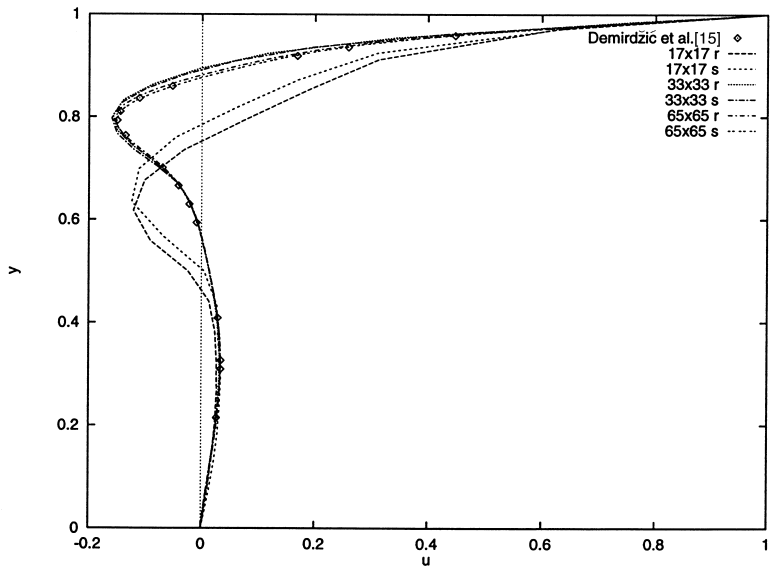


Figure 15. Centreline velocities (u) for the skewed lid-driven cavity.

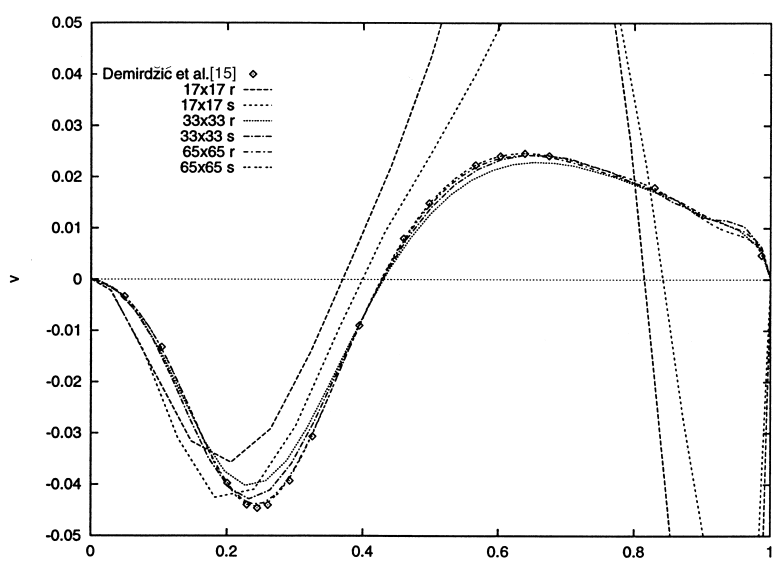


Figure 16. Centreline velocities (v) for the skewed lid-driven cavity.

grids, identified with the suffixed letters 's' and 'r' in the figure legends. Comparisons with simulations in Reference [15], where they used a 320×320 grid show good convergence.

4.3. Backward-facing step

Another extensively studied test case is flow over a backward-facing step (see, for instance, References [8,16,17]). The set-up we have used in our simulations, similarly to that in Reference [18,19], is depicted in Figure 17. The computational domain is the rectangle to the right of the step, with height H and length X_L. We choose $H = 1$ m, $h = H/2$ and $X_L = 27H$. The inflow is specified as a parabola with mean velocity $\bar{u}$. The other boundary conditions are $u_x = v_y = 0$ at the outflow, and the no-slip condition at the walls. The Reynolds number is defined as $Re = \bar{u}H/\nu$, and in our experiments we have chosen $Re = 400$.

In our first experiment we study how X_R, the length of the recirculating area, depends on the resolution. Figure 18 shows $u(x, 0.01H)$, the horizontal velocity component just above the bottom of the channel, for various uniform resolutions. For the 256×64 grid, $X_R = 4.25$ m $= 8.5h$, which is in agreement with Reference [19].

In order to test the local grid refinement procedure we used the three-level grid shown in Figure 19. Figure 20 compares $u(x, 0.01H)$ for locally refined grids with uniform grids corresponding to the finest resolutions. The figure shows almost no visible differences in the leftmost region, where the locally refined and uniform grids are equal. Elsewhere, the linear interpolation procedure used to determine $u(x, 0.01H)$ produces some differences.

5. SUMMARY

A multiblock/multilevel algorithm with local grid refinement applicable for general complex fluid flow has been presented. The discretization method using the SIMPLE method has been

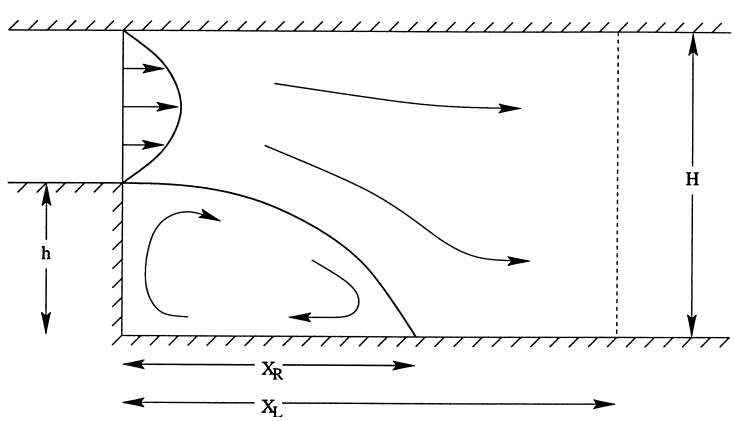


Figure 17. Backward-facing step.

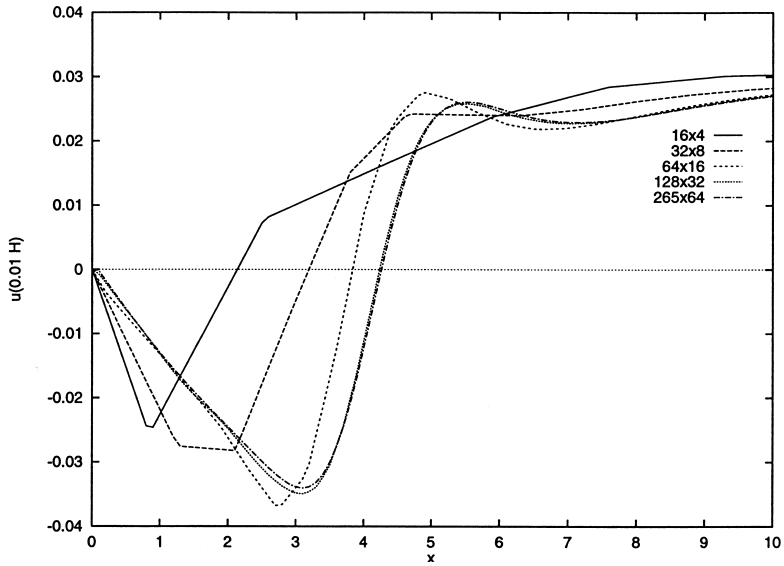


Figure 18. Horizontal velocity $0.01H$ above the bottom various uniform grids.

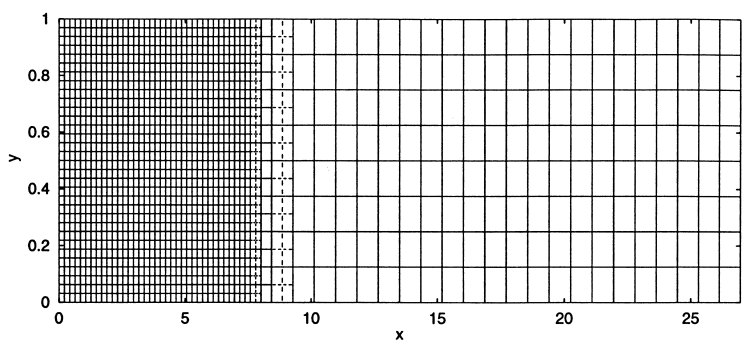


Figure 19. The three-level grid used for the calculations in Figure 20.

described. Details about the implementation of local grid refinement have been discussed. In particular, coarse grid flux correction and mass conservation are seen in connection with Dirichlet coupling and Neumann coupling between coarse and fine patches. Finally, numerical results for three standard test cases are given.

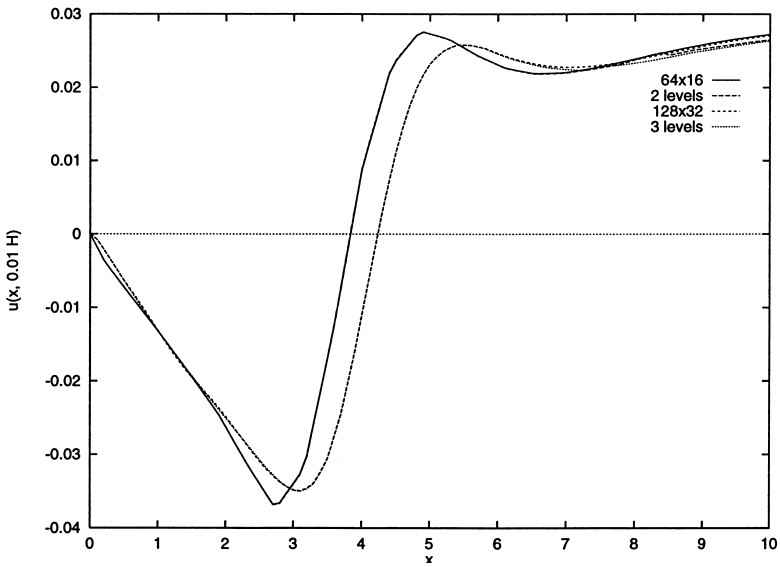


Figure 20. Horizontal velocity $0.01H$ above the bottom for locally refined and uniform grids.

REFERENCES

1. Burns AD, Wilkes NS. A finite difference method for the computation of fluid flows in complex three dimensional geometries. Technical Report HL87 1327(07), Harwell Laboratory, July, 1987.
2. Melaaen MC. Analysis of curvilinear non-orthogonal coordinates for numerical calculation of fluid flow in complex geometries. Dr ing thesis, NTH, Trondheim, Norway, 1990.
3. Rhie CM, Chow WL. Numerical study of the turbulent flow past an airfoil with trailing edge separation. *AIAA Journal* 1983; **21**(11): 1525–1532.
4. Patankar SV. *Numerical Heat Transfer and Fluid Flow*. Hemisphere: Washington, DC, 1980.
5. Deconinck H (ed.). *VKI Lecture Series*. von Karman Institute: Rhode St., Genese, Belgium, 1992.
6. Hackbusch W. *Iterative Solution of Large Sparse Systems of Equations*, Volume *95* of *Applied Mathematical Sciences*. Springer: Berlin, 1994.
7. Chen WL, Lien FS, Leschziner MA. Local mesh refinement within a multi-block structured-grid scheme for general flows. *Computer Methods in Applied Mechanics and Engineering* 1997; **144**: 327–369.
8. Shyy W, Thakur SS, Ouyang H, Liu J, Blosch E. *Computational Techniques for Complex Transport Phenomena*. Cambridge University Press: Cambridge, 1997.
9. Almgren AS, Bell JB, Colella P, Howell LH, Welcome ML. A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations. *Journal of Computational Physics* 1998; **142**(1): 1–46.
10. Berger MJ, Coella P. Local adaptive mesh refinement for shock hydrodynamics. *Journal of Computational Physics* 1997; **82**: 64–84.
11. Thompson CP, Leaf GK, Van Rosendale J. A dynamically adaptive multigrid algorithm for the incompressible Navier–Stokes equations—validation and model problems. *Applied Numerical Mathematics* 1992; **9**: 511–532.
12. Ferziger JH, Perić M. *Computational Methods for Fluid Dynamics*. Springer: Berlin, 1996.
13. Ghia U, Ghia KN, Shin CT. High-Re solutions for incompressible flow using the Navier–Stokes equations and a multigrid method. *Journal of Computational Physics* 1982; **48**: 387–411.
14. Thakur S, Shyy W, Udaykumar HS. Multiblock interface treatments in a pressure-based flow solver. *Numerical Heat Transfer Part B* 1998; **33**: 367–396.
15. Demirdžić I, Lilek Ž, Perić M. Fluid flow and heat transfer test problems for non-orthogonal grids: bench-mark solutions. *International Journal for Numerical Methods in Fluids* 1992; **15**: 329–354.

16. Armaly BF, Durst F, Pereira JCF, Schönung B. Experimental and theoretical investigation of backward-facing step flow. *Journal of Fluid Mechanics* 1983; **127**: 473–496.
17. Williams PT, Bakes J. Numerical simulations of laminar flow over a 3D backward-facing step. *International Journal for Numerical Methods in Fluids* 1997; **24**: 1159–1183.
18. Kim J, Moin P. Application of a fractional-step method to incompressible Navier–Stokes equations. *Journal of Computational Physics* 1985; **59**: 308–323.
19. Thompson MC, Ferziger JH. An adaptive multigrid technique for the incompressible Navier–Stokes equations. *Journal of Computational Physics* 1988; **82**: 94–121.

