Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Computational Physics 190 (2003) 572–600
JOURNAL OF COMPUTATIONAL PHYSICS
www.elsevier.com/locate/jcp

Gerris: a tree-based adaptive solver for the incompressible Euler equations in complex geometries

Stéphane Popinet *

National Institute of Water and Atmospheric Research, P.O. Box 14-901, Kilbirnie, Wellington, New Zealand

Received 10 September 2002; received in revised form 12 May 2003; accepted 2 June 2003

Abstract

An adaptive mesh projection method for the time-dependent incompressible Euler equations is presented. The domain is spatially discretised using quad/octrees and a multilevel Poisson solver is used to obtain the pressure. Complex solid boundaries are represented using a volume-of-fluid approach. Second-order convergence in space and time is demonstrated on regular, statically and dynamically refined grids. The quad/octree discretisation proves to be very flexible and allows accurate and efficient tracking of flow features. The source code of the method implementation is freely available.

AMS: 65M06; 65N06; 65N55; 76D05; 76-04

Keywords: Incompressible flow; Adaptive mesh refinement; Approximate projection method; Complex geometry

1. Introduction

Efficient techniques for the numerical simulation of low Mach number flows have a large range of applications: from fundamental fluid mechanics studies such as turbulence or interfacial flows, to engineering and environmental problems. For time-dependent flows, the finite speed of propagation of sound waves can lead to strong restrictions on the maximum value of the timestep. While filtering techniques can be applied to try to lift this constraint, a better approach is to assume that the fluid considered is strictly incompressible. This introduces an elliptic problem for the pressure which expresses the instantaneous propagation of pressure information throughout the entire domain. In practice, this leads to the fundamental change from a spatially explicit to a spatially implicit problem.

* Tel.: +64-4-386-0585; fax: +64-4-386-2153.
E-mail address: s.popinet@niwa.co.nz.

0021-9991/$ - see front matter
doi:10.1016/S0021-9991(03)00298-5

Projection methods and multigrid solvers have proved an efficient combination to solve this type of problem [1–4]. More recently, these techniques have been extended through the use of higher-order, unconditionally stable advection schemes [5,6].

Another characteristic of fluid flows is the very wide range of spatial scales often encountered: shocks in compressible flows, interfaces between immiscible liquids, turbulence intermittency, boundary layers and vorticity generation near solid boundaries are just a few examples. Consequently, in recent years a number of researchers have investigated the use of adaptive mesh refinement, where the spatial discretisation is adjusted to follow the scale and temporal evolution of flow structures [7–9].

For compressible flows, two main approaches have been developed: the hierarchical structured grid approach of Berger and Oliger (adaptive mesh refinement, AMR) [7] and quad/octree-based discretisations [8,10]. The AMR framework uses classical algorithms on regular Cartesian grids of different resolutions arranged hierarchically. The only modification necessary is to allow coupling between grids at different levels through the boundary conditions. Quad/octree discretisations, on the other hand, deal with various levels of refinement locally through the use of finite-difference operators adapted to work at fine/coarse cell boundaries.

The AMR framework has been extended to incompressible flows by Minion [11], Almgren et al. [12] and Howell and Bell [9] but we are not aware of any quad/octree implementation of adaptive mesh refinement for incompressible flows. The natural hierarchical nature of tree-based discretisations is well suited for multigrid implementations. Moreover, we believe that the flexibility and simplicity of mesh refinement and coarsening of quad/octrees can be a significant advantage when dealing with complex solid boundaries or evolving interfacial flows.

Complex solid boundaries are usually represented using boundary-following structured curvilinear grids or unstructured grids. While boundary conditions can be easily and accurately applied on such grids, grid generation can be a difficult and time consuming process. In recent years, "Cartesian grids" [13–17] and "immersed boundary" [18–20] techniques have known a regain of interest because they greatly simplify the grid generation process. This flexibility comes at the cost of a more complex treatment of boundary conditions at solid boundaries.

In this light, we present a numerical method for solving the incompressible Euler equations, combining a quad/octree discretisation, a projection method and a multilevel Poisson solver. Advection terms are discretised using the robust second-order upwind scheme of Bell et al. [5] and complex solid boundaries are treated through a Cartesian volume-of-fluid approach. On a uniform grid without solid boundaries, the approach presented reduces to the approximate projection method described by Martin (cf. [21,22]). Solid boundaries are treated using a combination of a Poisson solver similar to the one studied by Johansen and Colella (cf. [23,24]) and of a cell-merging technique for the advection scheme [14]. In contrast to classical AMR strategies, adaptive refinement is performed at the fractional timestep.

While we restrict this description to two-dimensional flows for clarity, the extension to three dimensions is straightforward: the source code of the three-dimensional parallel implementation [25] can be freely accessed, redistributed and modified under the terms of the Free Software Foundation General Public License.

2. Spatial discretisation

The domain is spatially discretised using square (cubic in 3D) finite volumes organised hierarchically as a quadtree (octree in 3D) [26]. This type of discretisation has been used and studied extensively for image processing and computer graphics applications [27,26] and more recently applied to the solution of the Euler equations for compressible flows [8,10]. An example of spatial discretisation and the corresponding tree representation is given in Fig. 1. In what follows we will refer to each finite volume as a *cell*. The length

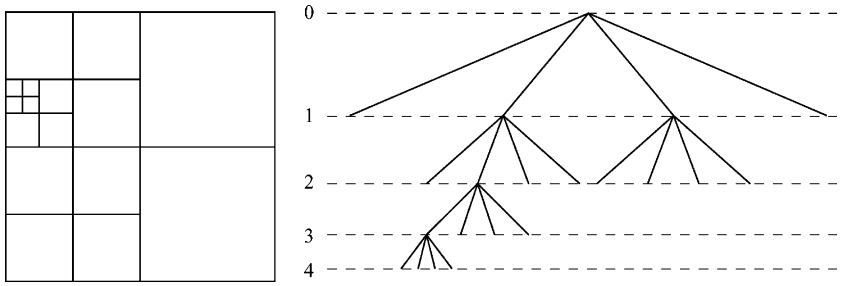

Fig. 1. Example of quadtree discretisation and corresponding tree representation.

of a cell edge is denoted by h. Each cell may be the *parent* of up to four *children* (eight in 3D). The *root cell* is the base of the tree and a *leaf cell* is a cell without any child. The *level* of a cell is defined by starting from zero for the root cell and by adding one every time a group of four descendant children is added. Each cell $\mathscr{C}$ has a *direct neighbour* at the same level in each direction d (four in 2D, six in 3D), noted $\mathscr{N}_d$. Each of these neighbours is accessed through a face of the cell, noted $\mathscr{C}_d$. In order to handle embedded solid boundaries, we also define *mixed cells* which are cut by a solid boundary.

To simplify the calculations required at the cell boundaries, we add the constraints illustrated in Fig. 2:

(a) the levels of direct neighbouring cells cannot differ by more than one;.

(b) the levels of diagonally neighbouring cells can not differ by more than one;

(c) all the cells directly neighbouring a mixed cell must be at the same level.

While not fundamentally necessary, these constraints greatly simplify the gradient and flux calculations presented in this article. Constraints (a) and (b) have little impact on the flexibility of the discretisation (they only impose gradual refinement by increments of two). Constraint (c) is more restrictive as it forces all the cells cut by the interface to be at the same level (i.e. the whole solid boundary must be described at the same resolution). It is also important to note that a major restriction of the quad/octree structure is that it imposes a locally spatially isotropic refinement. This can be an issue in highly non-isotropic flows (i.e., boundary layers, large scale atmospheric flows, etc.). A limited solution is to use a rectangle instead of a square as root cell, thus resulting in a fixed refinement ratio between the corresponding spatial directions. A more general (and complicated) approach would be to use the "variable quadtree" approach of Berger and Aftosmis [28].

In practice, the choice of a data structure to represent the tree is conditioned by the following requirements:

(a) for any given cell, efficient access to neighbouring cells;

(b) for any given cell, efficient access to cell level and spatial coordinates;

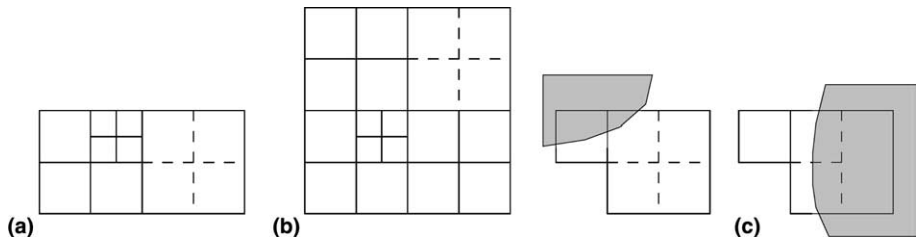

Fig. 2. Additional constraints on the quadtree discretisation. The refinement necessary to conform to the given constraint is indicated by the dotted lines.

(c) efficient traversal of:
- all leaf cells,
- all cells at a given level,
- all mixed cells.

At present, we use the *fully threaded tree* structure presented by Khokhlov [10] which allows (a) and (b) to be performed in $\mathcal{O}(1)$ operations (versus $\mathcal{O}(\log N)$ for a standard pointer-based structure). Operations (c) are performed in $\mathcal{O}(N \log N)$ using the standard pointer-based tree description (N is the number of cells traversed). Other modern quad/octree representations might be as good or better (in particular, the linear quadtree encoding of Balmelli et al. [29] is noteworthy).

The primitive variables of the Euler equations (velocity $\mathbf{U}$ and pressure p) are all defined at the centre of the cells. In mixed cells, the solid boundary is defined through a volume-of-fluid type approach. Specifically, we define:

- the volume fraction a as the ratio of the volume occupied by the fluid to the total volume of the cell;
- the surface fraction in direction d, s_d as the ratio of the area of face C_d occupied by the fluid to the total area of the face.

This solid boundary description assumes that the geometries represented do not possess features with spatial scales smaller than the mesh size. In particular, sharp angles or thin bodies cannot be represented correctly. This can be an issue for some applications, but more importantly, as argued by Day et al. [24], it will restrict the efficiency of the multigrid solver.

Computing the volume and area fractions can be expressed in terms of boolean operations (intersection, union, difference) between curves (in 2D) or volumes (in 3D). This is a difficult problem to solve in a robust manner (due to the limited precision of arithmetic operations in computers). Because of their numerous practical applications, robust geometrical operations have attracted considerable attention from the computational geometry community in recent years [30–33]. Drawing from these results, we use the boolean operations implemented in the GTS Library [34] based on an approach similar to that presented by Aftosmis et al. [35].

3. Temporal discretisation

We consider a constant density, incompressible and inviscid fluid. Given a velocity field

$$\mathbf{U}(x,y,t) = (u(x,y,t), v(x,y,t)),$$

and a pressure field $p = p(x,y,t)$ defined at location (x,y) and time t, on some domain Ω with a solid wall boundary $\partial\Omega$, the incompressible Euler evolution equations for $\mathbf{U}$ are

$$\mathbf{U}_t = -u\mathbf{U}_x - v\mathbf{U}_y - \nabla p,$$

$$\nabla \cdot \mathbf{U} = 0.$$

The boundary condition for the velocity at solid wall boundaries is the no-flow condition

$$\mathbf{U}(x,y,t) \cdot \mathbf{n} = 0 \quad \text{for } (x,y) \in \partial\Omega,$$

where $\mathbf{n}$ is the outward unit vector on $\partial\Omega$.

We use a classical fractional-step projection method [1,2,36]. At any given timestep n, we assume that the velocity at time n, $\mathbf{U}^n$ and the fractional step pressure $p^{n-1/2}$ are known at cell centres. In a first step, a provisional value $\mathbf{U}^{\star\star}$ is computed using

$$\frac{\mathbf{U}^{\star\star} - \mathbf{U}^n}{\Delta t} = -\mathbf{A}^{n+1/2}, \tag{1}$$

where $\mathbf{A}^{n+1/2}$ is an approximation to the advection term $[(\mathbf{U}\cdot\nabla)\mathbf{U}]^{n+1/2}$. The new velocity $\mathbf{U}^{n+1}$ is then computed by applying an approximate projection operator to $\mathbf{U}^{\star\star}$ which also yields the fractional step pressure $p^{n+1/2}$.

4. Poisson equation

The projection method relies on the Hodge decomposition of the velocity field as

$$\mathbf{U}^{\star\star} = \mathbf{U} + \nabla\phi, \tag{2}$$

where

$$\nabla\cdot\mathbf{U} = 0 \quad \text{in } \Omega \quad \text{and} \quad \mathbf{U}\cdot\mathbf{n} = 0 \quad \text{on } \partial\Omega. \tag{3}$$

Taking the divergence of (2) yields the Poisson equation

$$\nabla^2\phi = \nabla\cdot\mathbf{U}^{\star\star}, \tag{4}$$

while the normal component of (3) yields the boundary condition

$$\frac{\partial\phi}{\partial\mathbf{n}} = \mathbf{U}^{\star\star}\cdot\mathbf{n} \quad \text{on } \partial\Omega.$$

The divergence-free velocity field is then defined as

$$\mathbf{U} = \mathbf{U}^{\star\star} - \nabla\phi,$$

where ϕ is obtained as the solution of the Poisson problem (4). This defines the *projection* of the velocity $\mathbf{U}^{\star\star}$ onto the space of divergence-free velocity fields.

In the context of the approximate projection method we are using here the discrete formulation of the projection operator will depend on where the velocity field is discretised relative to the pressure field. We will use both an exact projection for face-centred advection velocities and an approximate projection for the final projection of the cell-centred velocities. The detail of these two projections does not influence the general description of the Poisson solver.

4.1. Relaxation operator

In practice, the spatially discretised Poisson problem results in a linear system of equations with the pressure at cell centres as unknowns

$$\mathscr{L}(\phi) = \nabla\cdot\mathbf{U}^{\star\star}, \tag{5}$$

where $\mathscr{L}$ is a discretisation of the Laplacian. This system can be solved through iterative methods (Jacobi, Gauss–Seidel) using a *relaxation operator*.

If we consider a discretisation cell $\mathscr{C}$ of boundary $\partial\mathscr{C}$, using the divergence theorem, the integration of (4) yields

$$\int_{\partial\mathscr{C}} \nabla\phi\cdot\mathbf{n} = \int_{\mathscr{C}} \nabla\cdot\mathbf{U}^{\star\star}, \tag{6}$$

where $\mathbf{n}$ is the outward unit normal of $\partial\mathscr{C}$. In the case of a cubic discretisation cell, the discrete equivalent of (6) can be written as

$$\sum_d s_d \nabla_d \phi = ha\nabla \cdot \mathbf{U}^{\star\star}, \tag{7}$$

where d is the direction, s_d the surface fraction in direction d and a the fluid volume fraction of the cell. Johansen and Colella [23] have shown that this discretisation is second-order accurate if the right-hand side is defined at the geometric centre of the partial cell and the gradient at the geometric centre of the partial faces. Expressing the gradient at the geometric centre of the partial face requires interpolation of the full-face-centered gradients. While this is relatively simple on a regular Cartesian grid, this is more difficult within the adaptive framework we are using. Consequently we have chosen to use the full-face-centered gradient even in mixed cells. The following description thus applies to both full and mixed cells.

To construct the relaxation operator, we assume that the face gradient can be expressed as a linear function of the pressure at the centre of the cell

$$\nabla_d \phi = \alpha_d \phi + \beta_d,$$

where the α are constants and the β are linear functions of the values of the pressure in the adjacent discretisation cells.

In practice, three cases must be considered for the construction of the gradient operator (Fig. 3). If the neighbour of the cell in direction d, $\mathscr{N}_d$ is at the same level and is a leaf cell, the gradient is simply $\nabla_d \phi = (\phi_d - \phi)/h$, where ϕ_d is the value of ϕ at the centre of $\mathscr{N}_d$. Using the notation above: $\alpha_d = -1/h$ and $\beta_d = \phi_d/h$.

Fig. 4 illustrates the case where $\mathscr{N}_d$ is at a lower level (case 3(b)). In order to maintain the second-order accuracy of the gradient calculation, it is necessary to use a three-point interpolation procedure. The gradient $\nabla_d \phi$ is computed by fitting a parabola through points ϕ_6, ϕ and either ϕ_7 or $\widehat{\phi}_d$. By construction, $\widehat{\mathscr{N}}_d$ is at the same level as $\mathscr{C}$. If $\widehat{\mathscr{N}}_d$ is a leaf cell, $\nabla_d \phi$ can be expressed as

$$h\nabla_d \phi = -\frac{\phi}{3} - \frac{\widehat{\phi}_d}{5} + \frac{8}{15}\phi_6, \tag{8}$$

where the value of the pressure at the centre of $\widehat{\mathscr{N}}_d$, $\widehat{\phi}_d$ has been used. If $\widehat{\mathscr{N}}_d$ is not a leaf cell, an interpolated value for the pressure ϕ_7 is constructed by averaging the values of its children closest to $\mathscr{C}$ (indicated by $\bigcirc$ in Fig. 4). The gradient is then given by

$$h\nabla_d \phi = -\frac{2}{9}\phi - \frac{8}{27}\phi_7 + \frac{14}{27}\phi_6. \tag{9}$$

The pressure ϕ_6 must itself be interpolated from ϕ_d and from the values in the neighbouring cells in directions perpendicular to d. Due to the *corner refinement* constraint (Fig. 2(b)), these cells ($\widehat{\mathscr{N}}_{\perp d}$ and $\mathscr{N}_{\perp d}$)

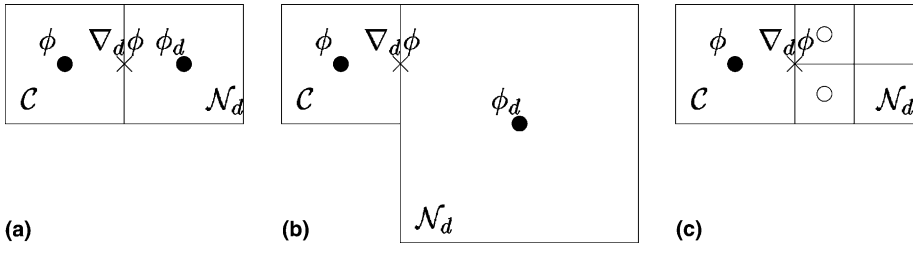

Fig. 3. Three cases for face-centered gradient calculation: (a) cells at the same level; (b) fine-coarse boundary; (c) coarse-fine boundary.

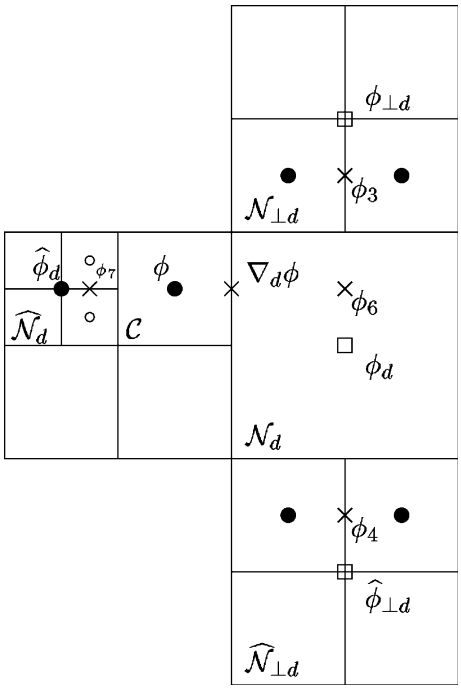

Fig. 4. Second-order interpolation used for the gradient calculation at fine/coarse cell boundaries.

are guaranteed to be at the same level as $\mathcal{N}_d$. The values ϕ_3 and ϕ_4 are derived using the same averaging procedure if $\widehat{\mathcal{N}}_{\perp d}$ and $\mathcal{N}_{\perp d}$ are not leaf cells. This leads to the following four cases:

$$\phi_6 = \begin{cases} \frac{15}{16}\phi_d - \frac{3}{32}\widehat{\boldsymbol{\phi}}_{\perp d} + \frac{5}{32}\phi_{\perp d} & \text{if } \widehat{\mathcal{N}}_{\perp d} \text{ and } \mathcal{N}_{\perp d} \text{ are leaf cells,} \\ \frac{5}{6}\phi_d - \frac{1}{14}\widehat{\boldsymbol{\phi}}_{\perp d} + \frac{5}{21}\phi_3 & \text{if } \widehat{\mathcal{N}}_{\perp d} \text{ is a leaf cell,} \\ \phi_d - \frac{1}{7}\phi_4 + \frac{1}{7}\phi_{\perp d} & \text{if } \mathcal{N}_{\perp d} \text{ is a leaf cell,} \\ \frac{8}{9}\phi_d - \frac{1}{9}\phi_4 + \frac{2}{9}\phi_3 & \text{otherwise} \end{cases} \tag{10}$$

The gradient $\nabla_d\phi$ can still be expressed as a linear function of ϕ. The corresponding values of α_d and β_d can be calculated by using (8)–(10).

In the third case, $\mathcal{N}_d$ is at the same level but is not a leaf cell (Fig. 3(c)). The gradient is simply constructed as minus the average of the gradients constructed from the children cells of $\mathcal{N}_d$ closest to $\mathcal{C}$ (indicated by $\bigcirc$ in Fig. 3(c)). These gradients are in turn computed using the interpolation technique described above (case 3(b)). This approach ensures that the pressure gradient fluxes across coarse/fine boundaries are consistent. The extension to three dimensions is straightforward.

Once the α and β coefficients have been computed for each cell face of the domain, using (7) a relaxation operator can be defined as

$$\mathscr{R}(\phi, \nabla\cdot\mathbf{U}^{\star\star}) : \phi \leftarrow \frac{ha\nabla\cdot\mathbf{U}^{\star\star} - \sum_d s_d\beta_d}{\sum_d s_d\alpha_d}. \tag{11}$$

In the case where all the cells are on the same level and there are no solid boundaries (regular Cartesian grid), the operator reduces to the classical stencil

$$\mathscr{R}(\phi, \nabla \cdot \mathbf{U}^{\star\star}) : \phi \leftarrow \frac{\sum_d \phi_d - h^2 \nabla \cdot \mathbf{U}^{\star\star}}{n},$$

where n is the number of directions (4 in 2D, 6 in 3D).

This operator, together with the interpolation procedure described above, has several desirable properties. It is second-order accurate in space at coarse/fine cell boundaries and uses a consistent flux estimation. In the case of cells cut by solid boundaries, the flux calculation is only first-order accurate in space, however.

4.2. Boundary conditions

Cells on the boundary of the domain or mixed cells may not have neighbours in all directions. If values for the pressure ϕ_d are required in one of these directions, either by the gradient operator $\nabla_d \phi$ or by the interpolation formula (10), they are set as equal to ϕ (the pressure at the centre of the cell considered). For cells entirely contained within the fluid, this is equivalent to a classical second-order accurate implementation of Neumann boundary conditions for the pressure.

4.3. Multilevel acceleration

The point relaxation defined by $\mathscr{R}$ can be accelerated using a multigrid technique [3,4]. When using quad/octrees, different choices are possible for the construction of the multilevel hierarchy. We have chosen to define a multilevel $\mathscr{M}_l$ of depth l as the set of cells $\mathscr{C}$ which satisfy either of the conditions:

- level of $\mathscr{C}$ is equal to l;
- $\mathscr{C}$ is a leaf cell of level smaller than l.

An example of such a hierarchy is given in Fig. 5. This is probably not the best possible hierarchy for multigrid acceleration, in the sense that not all cells get coarser when moving from one level to the next. It is relatively easy to manually generate a possibly better hierarchy such as illustrated in Fig. 6. However, the systematic generation of such optimised hierarchies involves a set of rules substantially more complicated than the two conditions given above. In practice, if the simple rules are used, the traversal of the cells belonging to $\mathscr{M}_l$ is straightforward to implement when using a pointer-based quad/octree structure.

Using this multilevel hierarchy, we apply a classical multigrid "V-cycle" using the *correction form* of the linear system (5)

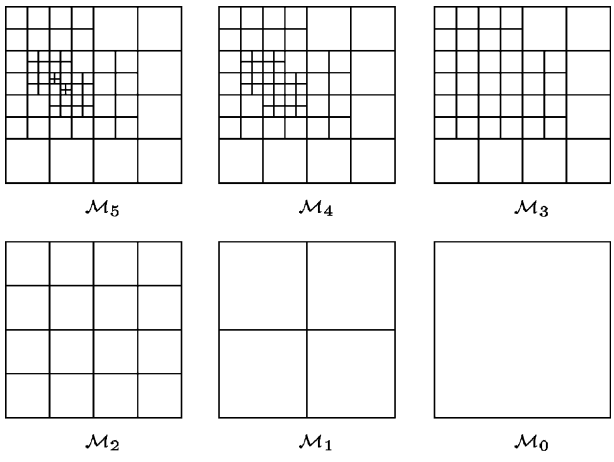

Fig. 5. Example of simple multilevel hierarchy.

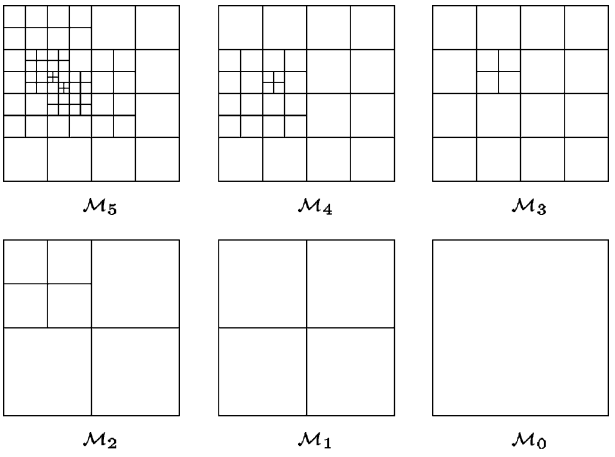

Fig. 6. Example of optimised multilevel hierarchy.

$$\mathscr{L}(\phi + \delta\phi) = \nabla \cdot \mathbf{U}^{\star\star} \iff L(\delta\phi) = R \quad \text{with } R = \nabla \cdot \mathbf{U}^{\star\star} - \mathscr{L}(\phi).$$

The residual R is first computed on all the cells of the deepest level $\mathscr{M}_L$ as

$$R_L = \nabla \cdot \mathbf{U}^{\star\star} - \frac{1}{ha}\sum_d s_d \nabla_d \phi.$$

The residual is then transferred recursively on all the coarser levels as a volume weighted average

$$R_l = \frac{\sum_i ah^2 R_{l+1}}{\sum_i ah^2},$$

where $\sum_i$ designates the summation over all the children of the cell considered. The value of the pressure correction $\delta\phi$ is then computed exactly on the coarsest level. This value is used as the initial guess on the next finer level. Straight injection is used i.e. the initial guess $\delta\phi$ in each cell of $\mathscr{M}_l$ is set as the value of $\delta\phi$ in its parent cell. The relaxation operator $\mathscr{R}$ is then applied a few times (using Jacobi iterations) and the resulting solution is used as initial guess on the next finer level. This is repeated recursively down to level L where the resulting correction is applied to ϕ. The whole V-cycle is repeated until the residual on the finest level is suitably small. This algorithm can be summarised as:

Compute R_L on $\mathscr{M}_L$
while $\|aR_L\|_\infty > \epsilon$
  for $l = L - 1$ **to** 0
    Compute R_l using weighted average of R_{l+1}
  end for
  Apply relaxation operator $\mathscr{R}(\delta\phi, R_0)$ to $\mathscr{M}_0$ down to convergence
  for $l = 1$ **to** L
    Get initial guess for $\delta\phi$ in cells at level l using straight injection from level $l - 1$
    Apply r times relaxations $\mathscr{R}(\delta\phi, R_l)$ to $\mathscr{M}_l$
  end for
  Correct ϕ on $\mathscr{M}_L$ using $\delta\phi$
  Compute R_L on $\mathscr{M}_L$
end while

It is important to note that, when applied to level $\mathscr{M}_l$, the relaxation operator should not use any cell of level larger than l (on which the solution for $\delta\phi$ is not yet defined). More specifically, when computing the gradient operator as described in the previous section, all the cells at level l must be considered as leaf cells even if they have children at level $l+1$.

This multigrid algorithm also differs from a classical implementation where a pre-relaxation is applied before transferring the residual onto the coarser level [3]. In a classical multigrid the solution computed at each level is thus a correction to the correction at a deeper level. Such a scheme is difficult to implement on the multilevel quadtree hierarchy illustrated in Figs. 5 and 6 because, depending on the way the refined patches are laid out, it would require the storage of multiple corrections for the cells used as boundary conditions for refined patches. The scheme we propose solves this problem by dealing on all levels only with the correction to the pressure on the finest level. Of course, the convergence rate of such a "half" V-cycle is less than the convergence rate of the classical version, but tests have shown that the increased speed of such a simplified V-cycle more than compensate for the decrease in convergence rate.

In the following, we generally stop the V-cycle iterations when the maximum volume-weighted residual $\|aR_L\|_\infty$ is smaller than 10^{-3} and we apply $r=4$ iterations of the relaxation operator at each level.

4.4. Numerical validation

We are interested in two main properties of the multilevel Poisson solver: the speed of convergence for each V-cycle iteration and the spatial order of the method as the grid is refined. Given the way the relaxation operator is constructed, the method is expected to be globally second-order accurate on both regular and refined grids. If solid boundaries are used, the method should be first-order accurate near the solid boundaries and second-order accurate elsewhere.

We define the volume-weighted norm of a variable e as

$$\|ae\|_p = \frac{\sum_i |e_i|^p a_i h^2}{\sum_i a_i h^2}, \tag{12}$$

where $\sum_i$ designates the summation over all the leaf cells of the domain. An ∞-norm, $\|ae\|_\infty$, is the maximum over all the leaf cells of the absolute value of e. Knowing two solutions defined on domains of maximum refinement L_1 and L_2, the rate of convergence in a given norm p can be estimated as

$$O_p = \frac{\log\left(\frac{\|e_1\|_p}{\|e_2\|_p}\right)}{(L_2 - L_1)\log 2}. \tag{13}$$

The convergence rate, $O_p = n$, indicates nth-order accuracy, i.e., the leading term in the truncation error scales as $\mathscr{O}(h^n)$.

A first test illustrates convergence on a regular Cartesian grid for a smooth pressure solution. We consider a square domain of size unity centred on the origin, with Neumann boundary conditions on all sides. The divergence is set in each cell as

$$\nabla \cdot \mathbf{U}^{\star\star}(x,y) = -\pi^2(k^2+l^2)\sin(\pi k x)\sin(\pi l y) \tag{14}$$

with $k = l = 3$. The exact solution of the Poisson equation with this source term is

$$\phi(x,y) = \sin(\pi k x)\sin(\pi l y) + \kappa, \tag{15}$$

where κ is an arbitrary constant. The initial guess for the pressure is a constant field. Seven levels of refinement are used which results in a Cartesian discretisation of $2^7 \times 2^7 = 128 \times 128$. We apply 10 iterations of the V-cycle with $r=4$ iterations of the relaxation operator at each level. Fig. 7 illustrates the evolution of

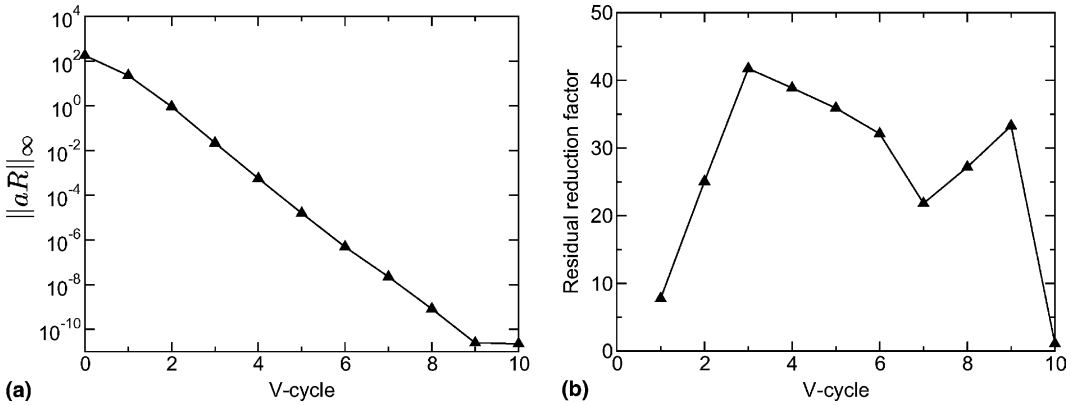

Fig. 7. Speed of convergence of the Poisson solver for a simple problem, $L = 7$: (a) evolution of the residual; (b) reduction factor.

the maximum norm of the residual. A reduction factor (ratio of the residuals before and after the V-cycle) of about 25 per V-cycle is obtained.

To estimate the order of the solver, we solved the same problem on regular grids of increasing resolution. For each grid size, the norm of the error on the solution is calculated using the computed solution and the exact solution given by (15), where κ is taken as the average value of the computed pressure over the entire domain. Fig. 8 illustrates the evolution of the error as a function of the depth of refinement L (i.e., a regular Cartesian grid of size $2^L \times 2^L$). The order of convergence is computed as indicated above. As expected for this simple problem, the method shows second-order convergence in all norms.

For the moment, only the classical stencil on regular meshes has been used. In order to test the accuracy of the gradient operator in the case of coarse/fine mesh boundaries, we use the following test. A domain is first discretised with $L - 2$ levels of refinement. Two more levels are then added only in the cells contained within a circle centred on the origin and of radius 1/4. The resulting discretisation for $L = 6$ is illustrated in Fig. 9. The same simple problem is then solved on this mesh. Fig. 10 gives the convergence rate of the residual for a mesh with $L = 7$. The residual reduction factor is about 15 per V-cycle. The order of the solver for the same problem is illustrated in Fig. 11. Close to second-order convergence in all norms is

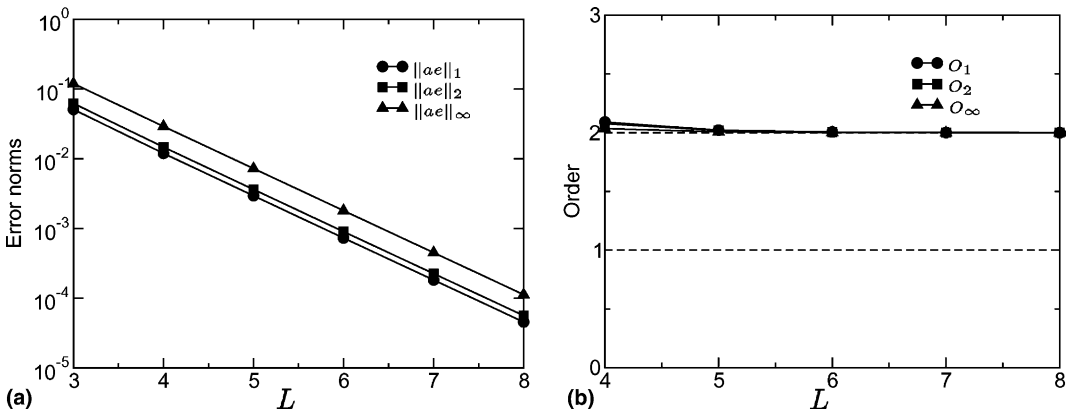

Fig. 8. Order of convergence of the Poisson solver for a simple problem: (a) evolution of the error and (b) order of convergence as functions of resolution.

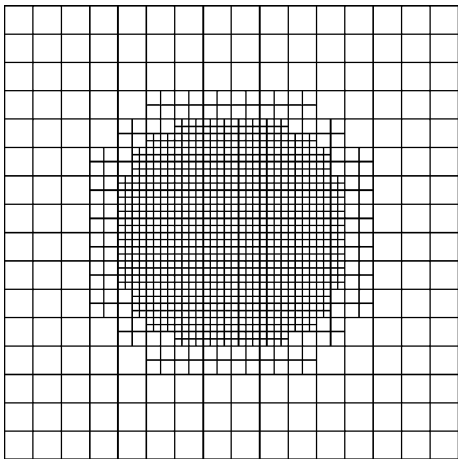

Fig. 9. Mesh used for evaluation of the coarse/fine gradient operators, $L = 6$.

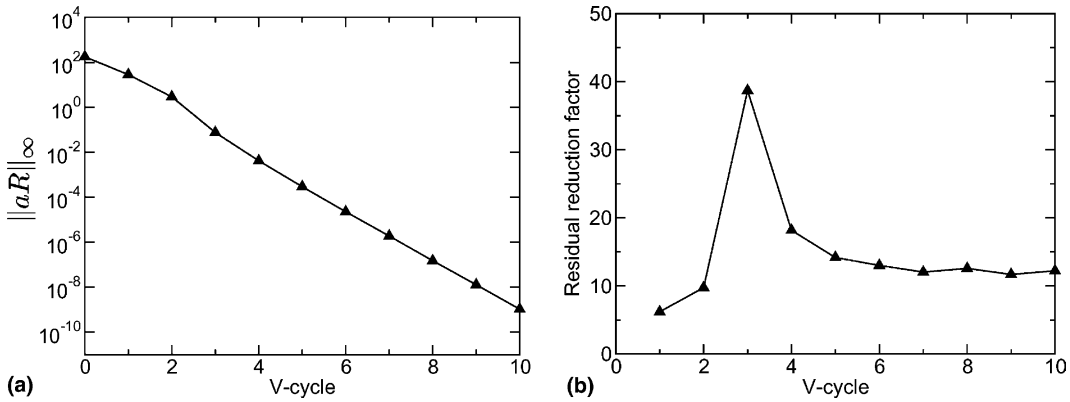

Fig. 10. Speed of convergence of the Poisson solver for a simple problem discretised using a mesh similar to Fig. 9 with $L = 7$: (a) evolution of the residual; (b) reduction factor.

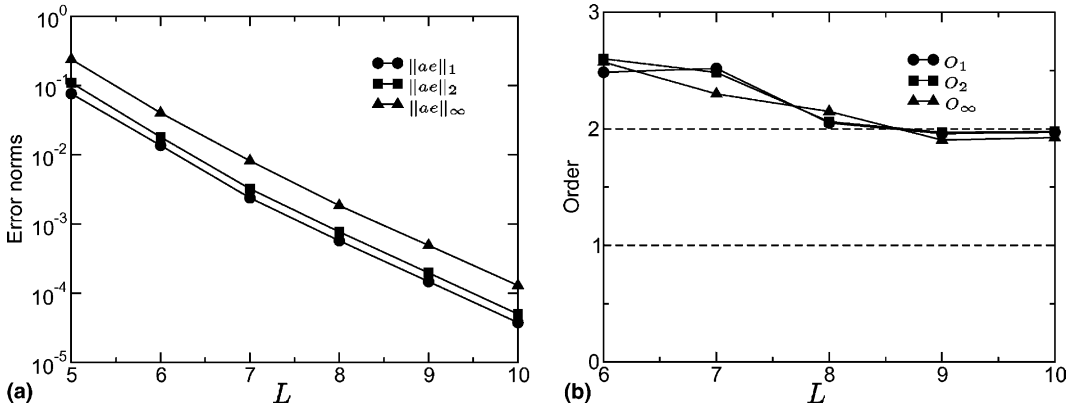

Fig. 11. Order of convergence of the Poisson solver for a simple problem discretised using a mesh similar to Fig. 9: (a) evolution of the error and (b) order of convergence as functions of resolution.

obtained which confirms that the gradient operator described previously is second-order accurate at coarse/fine mesh boundaries.

In order to test the ability of the method in presence of solid boundaries, we set up a series of tests with a variety of solid geometries. The corresponding solutions of the Poisson equation are illustrated in Fig. 12. All problems use the source term defined by (14) where x and y are the coordinates of the geometric centre of the cell considered [23] . A circular solid boundary centred on the origin and of radius 1/4 is used for (a). A star-shaped solid boundary defined in polar coordinates as

$$r(\theta) = 0.237 + 0.079\cos(6\theta)$$

is used in problem (b) and an ellipse centred on the origin measuring $\frac{3}{4} \times \frac{5}{8}$ in problem (c). All problems use Neumann conditions on all boundaries. Fig. 13 illustrates the convergence speed for the three problems with $L = 7$ levels of refinement. The "star" problem (b) is notably more difficult to solve with an average reduction factor of only five per V-cycle. This is due to the limitation of the volume-of-fluid representation of the solid boundaries. As mentioned earlier, the features of the solid boundaries are only represented accurately if their spatial scale is comparable to the mesh size. For the "star" problem, while the geometry is represented correctly on the finest level, it is not well represented on all the coarser levels used by the multigrid procedure. Cases (a) and (b) do not have this problem because the smallest spatial scales of the solid boundaries (circle and ellipse) are comparable to the domain size.

The evolution of the error with resolution and the associated convergence order is given in Fig. 14. As the exact solution of the problem is not known analytically, Richardson extrapolation is used. That is, the error for a given level of refinement L is computed by taking the solution at level $L + 1$ as reference.

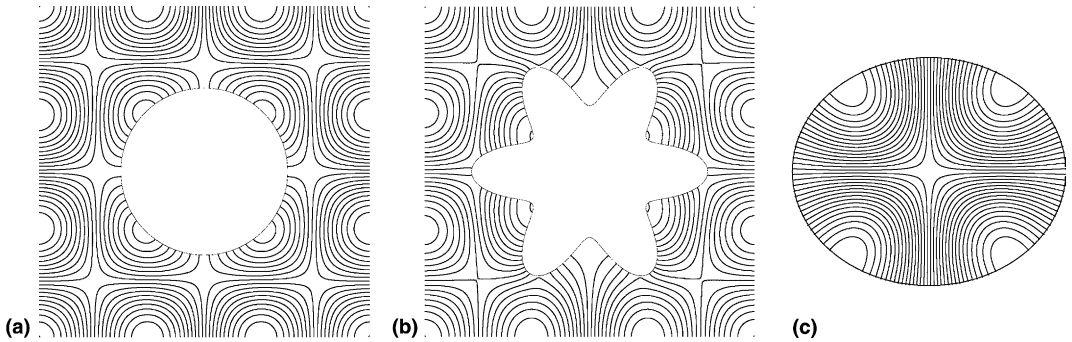

Fig. 12. Contour plots of the solution of Poisson problems with solid boundaries.

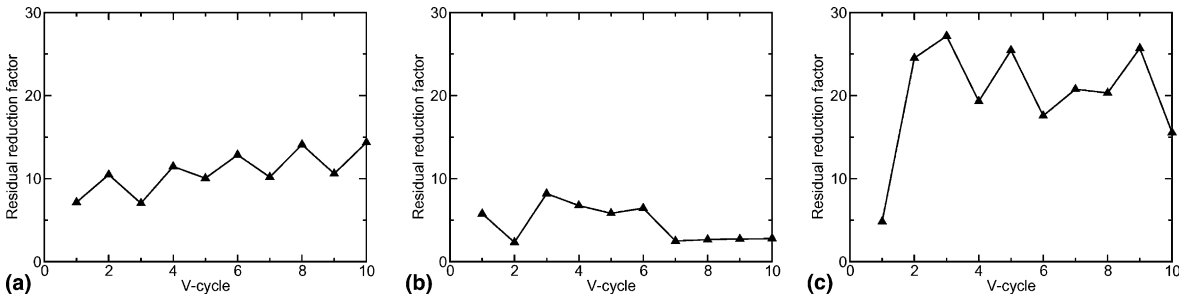

Fig. 13. Residual reduction factor for Poisson problems with solid boundaries, $L = 7$.

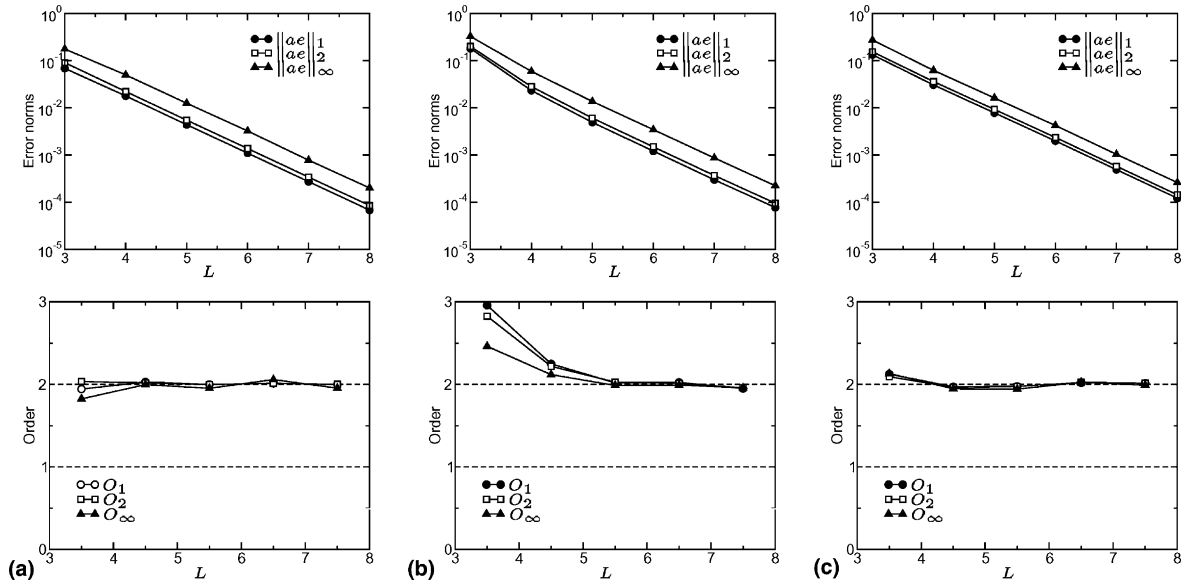

Fig. 14. Evolution of the error and associated convergence order for Poisson problems with solid boundaries.

A combination of solid boundaries and refinement is tested using a discretisation with $L-2$ levels of refinement on the whole domain plus two levels added only in cells cut by the solid boundary (a discretisation example is given in Fig. 15 for problem 12(b) and $L=6$). Figs. 16 and 17 illustrate the convergence speed and the order of the method using this discretisation.

The convergence is close to second-order (asymptotically in L) for all norms in all cases. The second-order convergence of the maximum error $\|ae\|_\infty$ is surprising as the discretisation of the pressure gradient

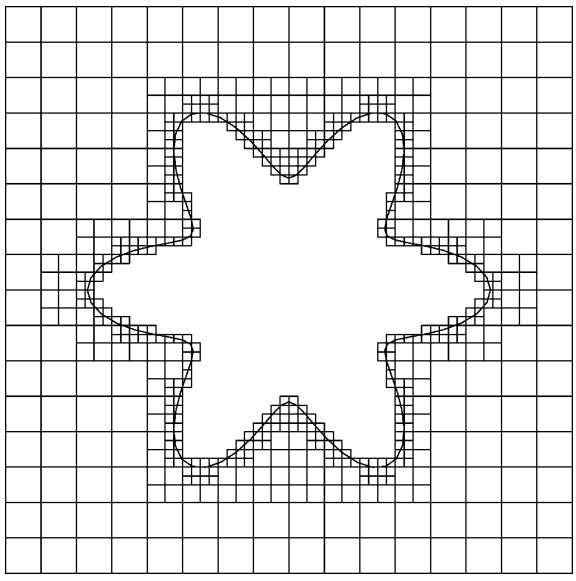

Fig. 15. Boundary-refined mesh for problem 12(b), $L=6$.

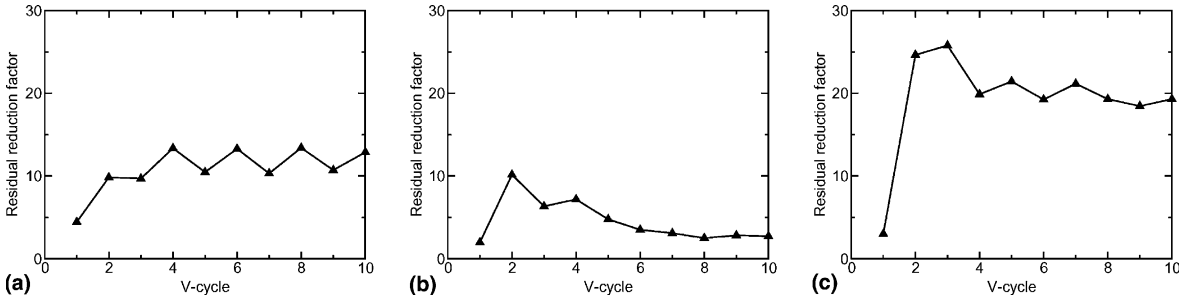

Fig. 16. Residual reduction factor for Poisson problems with refined solid boundaries, $L = 7$.

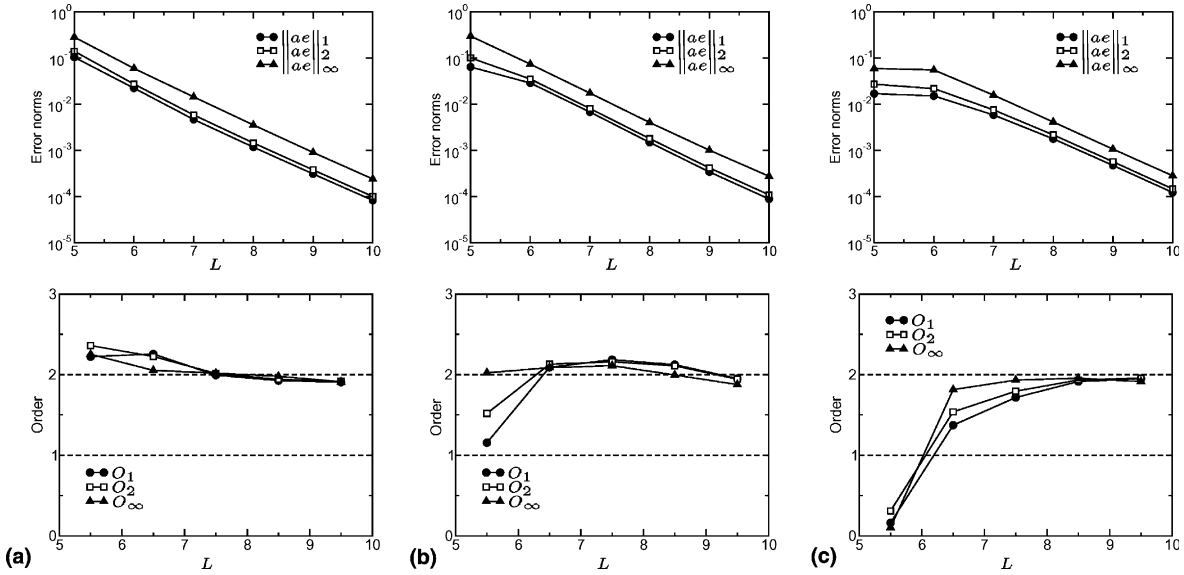

Fig. 17. Evolution of the error and associated convergence order for Poisson problems with refined solid boundaries.

fluxes is only first-order accurate near solid boundaries (as described in Section 4). This first-order error in the pressure gradient fluxes should lead to an $\mathcal{O}(1)$ truncation error of the Laplacian operator. Johansen and Colella [23] have demonstrated that a scheme with an $\mathcal{O}(1)$ truncation error will lead to an $\mathcal{O}(h)$ error on the solution for the pressure, in contradiction to the $\mathcal{O}(h^2)$ convergence we obtain here.

To try to clarify this issue we present truncation and solution errors for the test case used in [23,24]. The embedded boundary is defined by the curve

$$r(\theta) = 0.30 + 0.15\cos 6\theta.$$

The divergence is set in each full cell as

$$\nabla \cdot \mathbf{U}^{\star\star}(r,\theta) = 7r^2 \cos 3\theta.$$

The exaction solution for this system is $\phi(r,\theta) = r^4 \cos 3\theta$. A mesh similar to Fig. 15 is used, with two levels of refinement added near the embedded boundary. In mixed cells, in order to be able to use Neumann boundary conditions at the solid surface while retaining the exact solution, the flux of the gradient of the exact solution through the boundary is subtracted from the divergence, giving

$$\nabla \cdot \mathbf{U}^{\star\star}(r,\theta) = 7r^2\cos 3\theta - \frac{s(n_x\nabla_x\phi + n_y\nabla_y\phi)}{ah},$$

where s is the length of the embedded boundary contained within the cell, n_x and n_y are the components of the outward-pointing unit normal to the solid boundary. The gradients of the exact solution are defined as

$$\nabla_x\phi = \frac{4x^4 - 3x^2y^2 - 3y^4}{r}, \tag{16}$$

$$\nabla_y\phi = \frac{xy(5x^2 + 9y^2)}{r}, \tag{17}$$

where x and y are the coordinates of the center of mass of the piece of embedded boundary contained within the cell.

The results are summarized in Fig. 18. Fig. 18(a) gives the error norms and corresponding orders of convergence of the computed solution as functions of the level of refinement L. Fig. 18(b) illustrates the volume-weighted truncation error of the numerical Laplacian $\mathscr{L}$ defined in Section 4.1. As expected the max-norm of the truncation error of the numerical Laplacian is $\mathcal{O}(1)$ due to the $\mathcal{O}(h)$ error in the pressure gradient fluxes in mixed cells, while the orders of the 1- and 2-norm are close to one and one-half, respectively. However, while one would expect only first-order convergence of the max-norm of the error on the solution, second-order convergence in all norms is obtained as illustrated in Fig. 18(a). This confirms

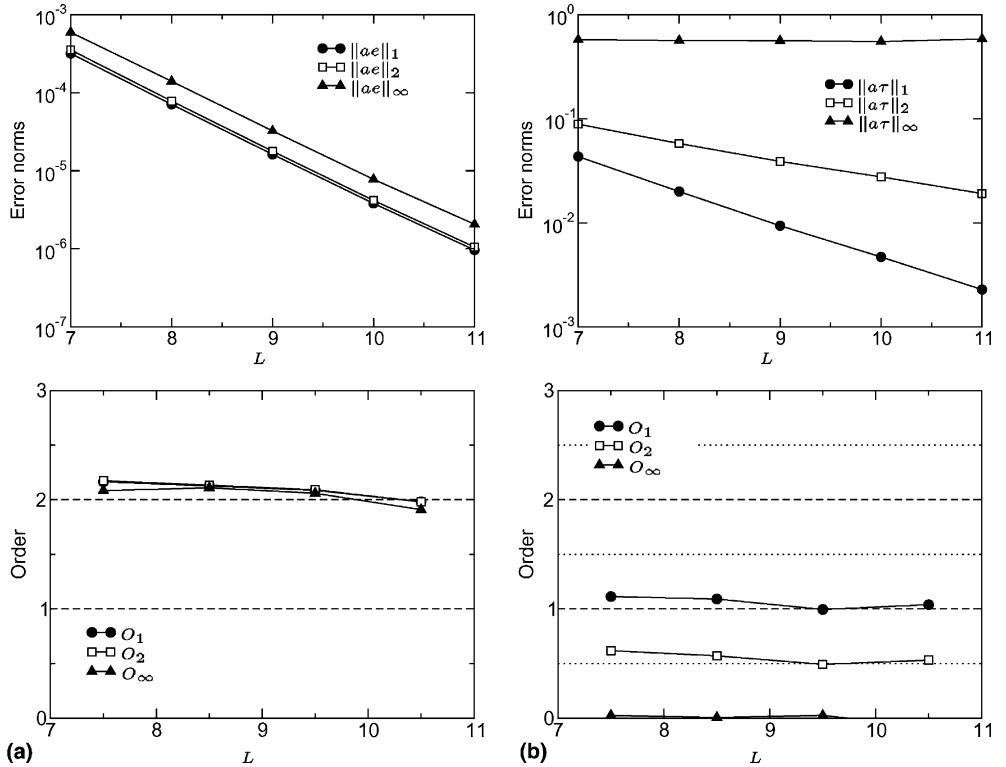

Fig. 18. Evolution of the error and associated convergence order for the Neumann Poisson problem of [23,24] using locally refined solid boundaries: (a) error on the solution; (b) volume-weighted truncation error on the Laplacian of the exact solution.

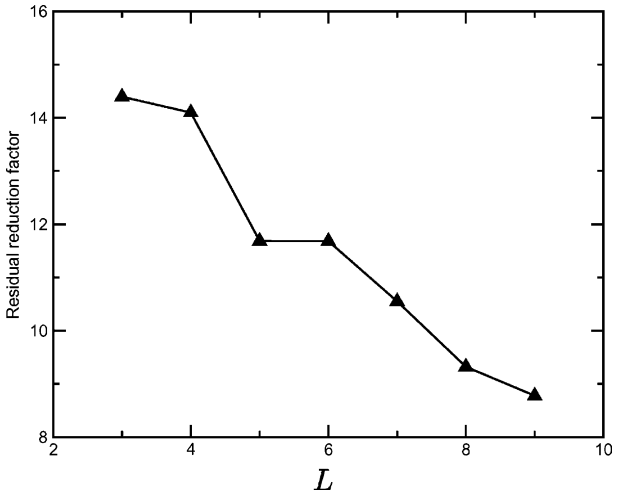

Fig. 19. Average residual reduction factor for problem 12(a) as a function of resolution L.

the results obtained for the previous tests and implies that second-order convergence in all norms can be obtained for practical problems even if the truncation error on the Laplacian is $\mathcal{O}(1)$. The discrepancy between our results and the theoretical study of Johansen and Colella could be explained if second-order converging errors in the bulk of the flow were always larger than first-order converging errors in mixed cells for all the tests we performed. This seems unlikely but if this were the case, it would be necessary to find more stringent test cases than used in this study or in [23,24]. Further work in this direction would be useful.

Finally, Fig. 19 shows how the solver scales with problem size. Problem 12(a) was solved on successively finer grids and the average residual reduction factor was computed as

$$\left(\frac{\|aR_0\|_\infty}{\|aR_n\|_\infty}\right)^{1/n},$$

where R_i is the residual after i V-cycle have been applied and n is the total number of V-cycles (10 in this test). The residual reduction factor decreases approximately linearly with resolution level L. The computational cost of solving a problem with $2^{2L} = N^2$ degrees of freedom (in 2D) thus scales as $\mathcal{O}(N^2 \log N)$ as expected from a multigrid scheme.

5. Advection term

We use a conservative formulation for the evaluation of the advection term. Given a cell $\mathscr{C}$ of boundary $\partial\mathscr{C}$, using the divergence theorem and the non-divergence of the velocity field, the finite-volume advection term $\mathbf{A}^{n+1/2}$ of (1) can be computed as

$$\int_{\mathscr{C}} \mathbf{A}^{n+1/2} = \int_{\mathscr{C}} [(\mathbf{U}\cdot\nabla)\mathbf{U}]^{n+1/2} = \int_{\mathscr{C}} [\nabla\cdot(\mathbf{U}\mathbf{U})]^{n+1/2} = \int_{\partial\mathscr{C}} \mathbf{U}^{n+1/2}(\mathbf{U}^{n+1/2}\cdot\mathbf{n}),$$

where $\mathbf{n}$ is the outward unit normal of $\partial\mathscr{C}$. In the case of our cubic discretisation cell this can be written

$$ah\mathbf{A}^{n+1/2} = \sum_d s_d \mathbf{U}_d^{n+1/2} u_d^{n+1/2}, \tag{18}$$

where $\mathbf{U}_d^{n+1/2}$ is the velocity at the centre of the face in direction d at time $n+1/2$ and $u_d^{n+1/2}$ is the normal component of the velocity at the centre of the face in direction d at time $n+1/2$. In order to compute these time- and face-centred values, we use a Godunov procedure [5], i.e., the leading terms of a Taylor series of the velocity of the form

$$\mathbf{U}_d^{n+1/2} = \mathbf{U}^n + \frac{h}{2}\partial_d \mathbf{U}^n + \frac{\Delta t}{2}\partial_t \mathbf{U}^n + \mathcal{O}(h^2, \Delta t^2),$$

where ∂_d designates the spatial derivative in direction d. Using the Euler equations, the temporal derivative can be replaced by spatial derivatives yielding

$$\mathbf{U}_d^{n+1/2} = \mathbf{U}^n + \left[\frac{h}{2} - \frac{\Delta t}{2}v_d^n\right]\partial_d \mathbf{U}^n - \frac{\Delta t}{2}v_{\perp d}^n \partial_{\perp d}\mathbf{U}^n - \frac{\Delta t}{2}\nabla p^n,$$

where $\perp d$ is the direction perpendicular to d in 2D (in 3D the sum over the two perpendicular directions) and v_d is the velocity component in direction d at the centre of the cell. Given a cell face, two values of $\mathbf{U}_d^{n+1/2}$ can be constructed, one for each cell sharing this face. In the original Godunov method for compressible fluids an unique value is constructed from these two values by solving a Riemann problem. In the incompressible case, simple upwinding is sufficient [5].

Following [21,22] we use a simplified upwind scheme of the form

$$\widetilde{\mathbf{U}}_d^{n+1/2}(\mathscr{C}) = \mathbf{U}^n + \frac{h}{2}\min\left(1 - v_d^n\frac{\Delta t}{h}, 1\right)\partial_d \mathbf{U}^n - \frac{\Delta t}{2}v_{\perp d}^n \overline{\partial}_{\perp d}\mathbf{U}^n, \tag{19}$$

where $\overline{\partial}_{\perp d}\mathbf{U}^n$ is the upwinded derivative in direction $\perp d$

$$\overline{\partial}_{\perp d}\mathbf{U}^n = \begin{cases} \nabla_{\perp d}\mathbf{U}^n & \text{if } v_{\perp d}^n < 0, \\ \nabla_{\widehat{\perp d}}\mathbf{U}^n & \text{if } v_{\perp d}^n > 0, \end{cases} \tag{20}$$

$\nabla_{\perp d}$ is computed as in Section 4.1 and $\widehat{\perp d}$ is the direction opposite to $\perp d$. The cell-centred derivative $\partial_d \mathbf{U}^n$ is computed by fitting a parabola through the centre of $\mathscr{C}$ and of its neighbours in directions d and $\widehat{d}$. If the neighbours are on different levels, an interpolation or averaging procedure similar to that presented in Section 4.1 is used. In the case of neighbouring cells at the same level, this procedure reduces to the classical second-order accurate centred difference scheme. We also do not use any slope limiters on the derivatives as we do not expect strong discontinuities in the velocity field for incompressible flows. Slope limiters can easily be added in this scheme if necessary.

Given the time- and face-centred values $\widetilde{\mathbf{U}}_d^{n+1/2}(\mathscr{C})$ and $\widetilde{\mathbf{U}}_{\widehat{d}}^{n+1/2}(\mathscr{N}_d)$ we then choose the upwind state

$$\mathbf{U}_d^{n+1/2}(\mathscr{C}) = \mathbf{U}_{\widehat{d}}^{n+1/2}(\mathscr{N}_d) = \begin{cases} \widetilde{\mathbf{U}}_d^{n+1/2}(\mathscr{C}) & \text{if } u_d^n > 0, \\ \widetilde{\mathbf{U}}_{\widehat{d}}^{n+1/2}(\mathscr{N}_d) & \text{if } u_d^n < 0, \\ \frac{1}{2}\left(\widetilde{\mathbf{U}}_d^{n+1/2}(\mathscr{C}) + \widetilde{\mathbf{U}}_{\widehat{d}}^{n+1/2}(\mathscr{N}_d)\right) & \text{if } u_d^n = 0, \end{cases} \tag{21}$$

where $\mathscr{N}_d$ is the neighbour of $\mathscr{C}$ in direction d. If $\mathscr{N}_d$ is at a lower level (Fig. 20) and $u_d^n \leqslant 0$, the value upwinded from $\mathscr{N}_d$ at the centre of the face (marked by $\circ$) is interpolated linearly from $\widetilde{\mathbf{U}}_{\widehat{d}}^{n+1/2}(\mathscr{N}_d)$ and from the value for its neighbour (or its children) in the correct direction, $\widetilde{\mathbf{U}}_{\widehat{d}}^{n+1/2}(\mathscr{N}_{\perp d})$.

In order to compute the advection term using (18), we first need to construct the face- and time-centred normal velocities $u_d^{n+1/2}$. If we want the method to be conservative, these normal velocities have to be discretely divergence-free. In a first step, normal velocities are constructed for both sides of each cell face using (19) and (20) where v_d^n and $v_{\perp d}^n$ are the corresponding components of the centred velocity $\mathbf{U}^n$. The upwind state $u_d^\star$ is then selected for each face using (21) where u_d^n is obtained by linear interpolation of the

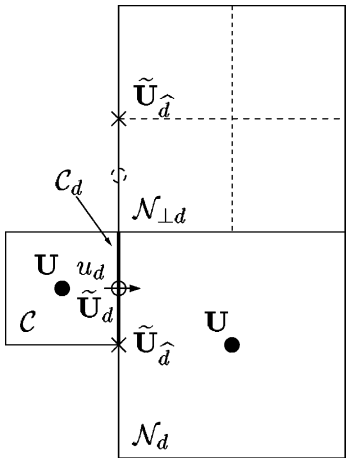

Fig. 20. Upwinding in the case of neighbouring cells at different levels. Linear interpolation is used to derive the value on the right side of $\mathscr{C}_d$.

relevant component of the centred velocities $\mathbf{U}^n(\mathscr{C})$ and $\mathbf{U}^n(\mathscr{N}_d)$. To make this set of normal velocities divergence-free we then apply a projection step by solving

$$\mathscr{L}(\phi) = \nabla \cdot \mathbf{u}^\star, \tag{22}$$

where $\nabla \cdot \mathbf{u}^\star$ is the finite-volume divergence of the normal velocity field, expressed for each cell as

$$\nabla \cdot \mathbf{u}^\star = \frac{1}{ah} \sum_d s_d u_d^\star. \tag{23}$$

By correcting $\mathbf{u}^\star$ with the pressure solution, we obtain a set of face- and time-centred normal velocities

$$u_d^{n+1/2} = u_d^\star - \nabla_d \phi. \tag{24}$$

When correcting the normal velocities, we also calculate a cell-centred value for the pressure gradient by simple averaging of face gradients

$$\nabla_d^\star \phi = \frac{\nabla_d \phi - \nabla_{\hat{d}} \phi}{2}. \tag{25}$$

To compute the advection term $\mathbf{A}^{n+1/2}$, we first need to re-predict the face-centred velocities $\mathbf{U}_d^{n+1/2}$, this time using $u_d^{n+1/2}$ rather than averages from cell-centred values. Again, in a first step, normal velocities are constructed for both sides of each cell face using (19) and (20) where we now take $v_d^n = (u_d^{n+1/2} - u_{\hat{d}}^{n+1/2})/2$. A unique value $\mathbf{U}_d^\star$ is then selected for each face using (21). A face-centred pressure gradient $\widetilde{\nabla}\phi$ is then computed by linear interpolation from the average cell-centred values $\nabla^\star\phi(\mathscr{C})$ and $\nabla^\star\phi(\mathscr{N}_d)$ (or its children). The predicted value is then obtained as

$$\mathbf{U}_d^{n+1/2} = \mathbf{U}_d^\star - \widetilde{\nabla}\phi.$$

Note that we could re-use the face- and time-centred normal velocities $u_d^{n+1/2}$ as predicted values (the tangential component would still need to be recalculated), however, we have found this approach to be unstable for flow around sharp angles. The spatial filtering of the pressure gradient provided by the averaging procedure seems to be necessary to ensure stability in this particular case.

5.1. Small-cell problem

To obtain the provisional cell-centred velocity field $\mathbf{U}^{\star\star}$ using (1), it is necessary to divide the finite-volume advection term (18) by the volume of the cell (ah^2) to get $\mathbf{A}^{n+1/2}$. This leads to the classical CFL stability condition

$$\frac{\|\mathbf{U}\|\Delta t}{ah} \leqslant 1,$$

which expresses the condition that a cell should not "overflow" during a given timestep. In the general case, the fluid fraction a can be arbitrarily small with a corresponding restrictive condition on the maximum timestep. This is traditionally referred to as the "small-cell problem". A number of approaches exist to work around this problem: cell merging [14,37], redistribution [15] or special difference schemes [13]. We have chosen to use a simple cell-merging technique similar to that presented by Quirk [14]. At initialisation time, after the volume and area fractions have been computed, all small cells are assigned a pointer to their biggest neighbour $\mathscr{B}$ (as measured by the fluid fraction a). For a given timestep, the advection term $ah^2\mathbf{A}^{n+1/2}$ is then computed in all cells as described above. To compute the advection update to the velocity, small cells are first grouped with adjacent mixed or full cells using the following recursive algorithm:

```
Group (𝒢, 𝒞)
  if 𝒞 does not already belong to any group then
    Add 𝒞 to 𝒢
    if 𝒞 is a small cell then
      Group (𝒢, ℬ(𝒞))
    end if
    for each direction d
      if 𝒩_d is a small cell then
        Group (𝒢, 𝒩_d)
      end if
    end for
  end if
```

where $\mathscr{G}$ is the resulting group of cells. For each group, the weighted averaged update is computed as

$$\mathbf{A}_{\mathscr{G}}^{n+1/2} = \frac{\sum_{\mathscr{G}} ah^2\mathbf{A}^{n+1/2}}{\sum_{\mathscr{G}} ah^2}.$$

Each cell in the group then receives a fraction of the update proportional to its volume

$$\mathbf{A}^{n+1/2} \approx \frac{ah^2}{\sum_{\mathscr{G}} ah^2}\mathbf{A}_{\mathscr{G}}^{n+1/2}.$$

This is equivalent to using a "virtual" cell formed by all the cells in the group. The CFL stability now depends on the total volume of the group of cells. In practice, choosing to define small cells as cells for which $a < 1/2$ ensured stability in all the cases we tested.

6. Approximate projection

While it is easy to formulate an exact projection operator for MAC (staggered, face-based) discretisation of the velocity field, it is difficult to do the same for a cell-centred discretisation. This is due to the spatial decoupling of the stencils used for the relaxation operator. This can cause numerical instabilities in the

pressure field and makes efficient implementation of multigrid techniques difficult [38,9]. Attempts to couple neighbouring pressure cells through asymmetric operators have been unsuccessful [39].

Drawing from these conclusions, Almgren et al. [12] dropped the requirement of exact discrete non-divergence of the projected cell-centred velocity field and proposed to use an approximate Laplacian operator well-behaved with respect to spatial coupling. Following Lai [38], Minion [11] and Martin [21] we use an approximate projection based on face-centred interpolation of the cell-centred velocity field. In a first step, face-centred normal components of the velocity are constructed by interpolation of the cell-centred provisional velocity $\mathbf{U}^{\star\star}$. This normal (MAC) velocity field is then projected using the exact projection operator (following steps (22) to (24)) and average cell-centred pressure gradients are constructed (using (25)). These pressure gradients are then used to correct $\mathbf{U}^{\star\star}$ to obtain the approximately divergence-free velocity field $\mathbf{U}^{n+1}$.

A detailed study of the stability of the approximate projection can be found in [40,41]. The use of pressure filters was found to be necessary in some cases (long, quasi-stationary simulations) to avoid a gradual build-up of non-divergence-free velocity modes. We do not use pressure filters in the current version of the code but did not encounter any noticeable numerical instabilities for the various tests we performed.

It is also important to note that even if the resulting cell-centred velocity field is not exactly divergence-free, the face-centred normal advection field $u_d^{n+1/2}$ is exactly discretely divergence-free, so that the advection scheme is exactly conservative. This is particularly important for the treatment of variable density flows.

7. Adaptive mesh refinement

Using a tree-based discretisation, it is relatively simple to implement a fully flexible adaptive refinement strategy.

In a first step, all the leaf cells which satisfy a given criterion are refined (as well as their neighbours when necessary, in order to respect the constraints described in Fig. 2). This step could be repeated recursively but we generally assume that the flow is evolving slowly (compared to the frequency of adaptation) so that only one pass is necessary.

In a second step, we consider the parent cells of all the leaf cells (i.e., the immediately coarser discretisation). All of these cells which do not satisfy the refinement criterion are coarsened (i.e., become leaf cells).

The values of the cell-centred variables for newly created or coarsened cells must be initialised. For newly coarsened cells, it is consistent to compute these values as the volume weighted average of the values of their (defunct) children, so that quantities such as momentum are preserved exactly. For newly created cells, the solution is less obvious. In particular, it is desirable that momentum and vorticity are locally preserved. Unfortunately, this is not simple to achieve in practice. We have chosen a simple linear interpolation procedure using the parent cell value and its gradients. Given a newly created cell $\mathscr{C}$ with parent cell $\mathscr{P}$, the new cell-centred value $v(\mathscr{C})$ is obtained as

$$v(\mathscr{C}) = v(\mathscr{P}) + \Delta_x \nabla_x v(\mathscr{P}) + \Delta_y \nabla_y v(\mathscr{P}),$$

where (Δ_x, Δ_y) are the coordinates of the centre of $\mathscr{C}$ relative to the centre of $\mathscr{P}$. This formula guarantees local conservation of momentum but tends to introduce numerical noise in the vorticity field. A better choice may be higher-order interpolants such as bicubic interpolation.

On the new discretisation, there is no guarantee that the velocity field is divergence-free anymore. A projection step is then needed. To avoid the cost of an extra projection step when adapting the grid, we perform the grid refinement at the fractional timestep, using the provisional velocity field $\mathbf{U}^{\star\star}$, just before the approximate projection is applied.

Various choices are possible for the refinement criterion. An attractive option would be to use Richardson extrapolation to obtain a numerical approximation of the truncation error of the whole scheme

[42,21]. For the moment, we use a simple criterion based on the norm of the local vorticity vector. Specifically, a cell is refined whenever

$$\frac{h\|\nabla \times \mathbf{U}\|}{\max \|\mathbf{U}\|} > \tau,$$

where $\max \|\mathbf{U}\|$ is evaluated over the entire domain. The threshold value τ can be interpreted as the maximum acceptable angular deviation (caused by the local vorticity) of a particle travelling at speed $\max \|\mathbf{U}\|$ across the cell.

The computational cost of this algorithm is small compared to the cost of the Poisson solver. It can be applied at every timestep with a negligible overall penalty (less than 5% of the total cost).

8. Numerical results

Following Minion [11] and Almgren et al. [43], we present two convergence tests illustrating the second-order accuracy of our method for flows without solid boundaries. The first problem uses a square unit domain with periodic boundary conditions in both directions. The initial conditions are taken as

$$u(x,y) = 1 - 2\cos(2\pi x)\sin(2\pi y),$$

$$v(x,y) = 1 + 2\sin(2\pi x)\cos(2\pi y).$$

The exact solution of the Euler equations for these initial conditions is

$$u(x,y,t) = 1 - 2\cos(2\pi(x-t))\sin(2\pi(y-t)),$$

$$v(x,y,t) = 1 + 2\sin(2\pi(x-t))\cos(2\pi(y-t)),$$

$$p(x,y,t) = -\cos(4\pi(x-t)) - \cos(4\pi(y-t)).$$

As in [43] nine runs are performed on grids with $L = 5, 6$ and 7 levels of refinement (labelled "uniform") and with one (labelled $r = 1$) or two (labelled $r = 2$) additional levels added only within the square defined by the points $(-0.25, -0.25)$ and $(0, 0)$. The length of the run for each case is 0.5, the CFL number is 0.75. For each run both the L_2 and L_∞ norms of the error in the x-component of the velocity is computed using (12) for both the whole domain (labelled "domain") and the refined region only (labelled "patch"). Table 1 gives the errors and order of convergence obtained.

Close to second-order convergence is obtained (asymptotically in L) for the L_2 and L_∞ norms on both uniform and refined domains. The values obtained are comparable to that in [11,43]. The error in the

Table 1
Errors and convergence orders in the x-component of the velocity for a simple periodic problem

	L_2					L_∞				
	$L = 5$	O_2	$L = 6$	O_2	$L = 7$	$L = 5$	O_∞	$L = 6$	O_∞	$L = 7$
Patch										
$r = 1$	6.80e−3	2.19	1.49e−3	2.05	3.61e−4	1.73e−2	1.82	4.89e−3	1.91	1.30e−3
$r = 2$	4.91e−3	1.66	1.55e−3	1.81	4.39e−4	1.58e−2	1.41	5.96e−3	1.84	1.66e−3
Domain										
Uniform	7.70e−3	2.87	1.05e−3	2.65	1.67e−4	1.74e−2	2.62	2.84e−3	2.68	4.44e−4
$r = 1$	9.52e−3	2.39	1.81e−3	2.17	4.01e−4	2.27e−2	2.14	5.15e−3	1.93	1.35e−3
$r = 2$	1.22e−2	2.19	2.67e−3	2.09	6.29e−4	2.76e−2	2.21	5.96e−3	1.84	1.66e−3

refined patch is comparable to the error at the resolution of the base grid. This is expected, given the arbitrary placement of the refined patch, the error is controlled mainly by the surrounding coarse cells.

The second test is the four-way vortex merging problem of Almgren et al. [43]. It demonstrates the convergence of the method when refinement is placed appropriately.

Four vortices are placed in the unit-square, centred at $(0,0)$, $(0.09,0)$, $(-0.045,0.045\sqrt{3})$ and $(-0.045,-0.045\sqrt{3})$ and of strengths -150, 50, 50, 50, respectively. The profile of each vortex centred around (x_i,y_i) is

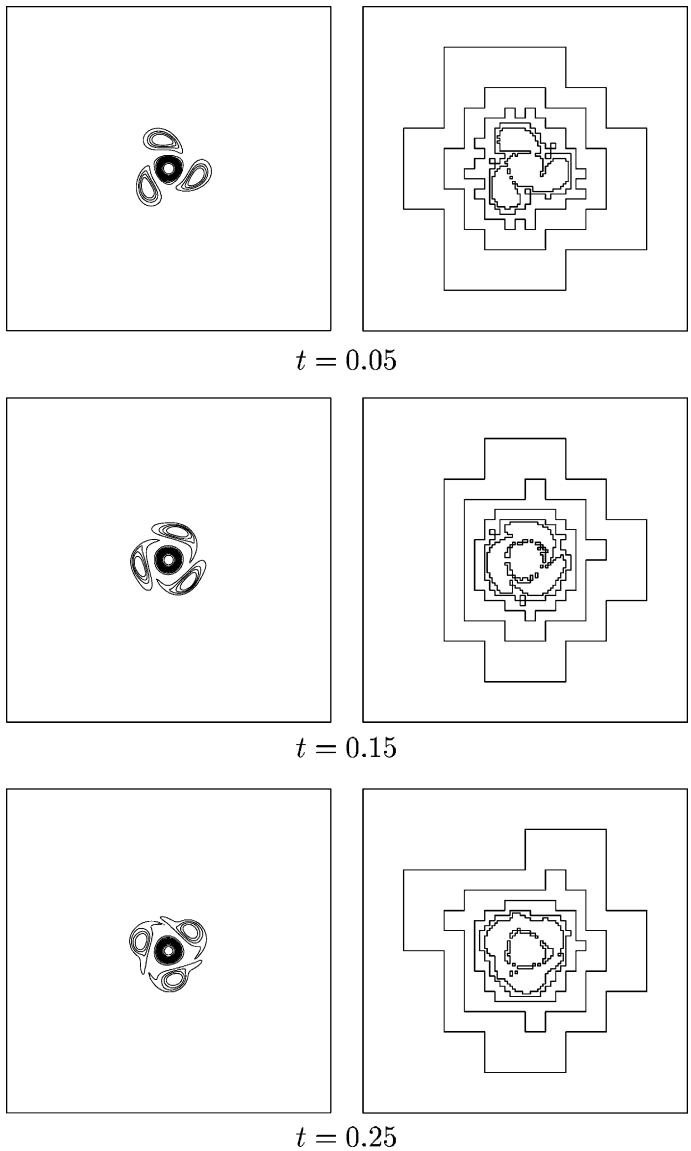

Fig. 21. Contour plots of vorticity (left) and adaptive grids used (right) for the four-way vortex merging calculation. The lines on the pictures in the right column represent the boundaries between levels of refinement (with a maximum of $L = 8$ levels).

$$\frac{1+\tanh(100(0.03-r_i))}{2},$$

where $r_i = \sqrt{(x-x_i)^2+(y-y_i)^2}$. To initialise the velocity field, we use this vorticity as the source term in the Poisson equation for the streamfunction ψ

$$\nabla^2\psi = \|\nabla\times\mathbf{U}\|.$$

Each component of the velocity field is then calculated from the streamfunction. No-flow boundary conditions are used on the four sides of the domain and the simulations are ran to $t = 0.25$ using a CFL of 0.9.

Five different discretisations are used, each time with up to L levels of refinement: a uniform grid, a grid using static refinement in concentric circles of decreasing radius and the dynamic adaptive refinement described in Section 7. The "circle" grid is constructed by starting from a uniform grid with four levels of refinement and by successively adding one level to all the cells contained within circles centred on the origin and of radii:

- $L = 6$: 0.25, 0.15;
- $L = 7$: 0.25, 0.2, 0.15;
- $L = 8$: 0.25, 0.2, 0.175, 0.15;
- $L = 9$: 0.25, 0.2, 0.175, 0.1625, 0.15;
- $L = 10$: 0.25, 0.225, 0.2, 0.175, 0.1625, 0.15.

For the dynamically refined grid, the vorticity-based criterion is applied at every timestep with a threshold $\tau = 4\times10^{-3}$. As we do not have an analytical solution for this problem, Richardson extrapolation is used.

Fig. 21 illustrates the evolution of the vorticity and of the adaptively refined grid for $L = 8$. The most refined level closely follows the three outer vortices as they orbit the central one. Far from the vortices, a very coarse mesh is used ($l = 3$). One may note a few isolated patches of refinement scattered at the periphery of the outer vortices. They are due to the numerical noise added to the vorticity by the interpolation procedure necessary to fill in velocity values for newly created cells. As mentioned in Section 7, this could be improved by using higher-order interpolants. This numerical noise is small enough that it does not compromise the convergence properties of the adaptive method (as shown below).

Table 2 summarises the results obtained for the first 12 calculations. For fine enough grids close to second-order convergence is obtained for both norms and for the three discretisations used. The norms of the error on the various grids are also comparable for a given resolution.

Table 3 gives the CPU time and the size of the problems solved for all three grids and for two levels of refinement. A PC-compatible Pentium 350 MHz machine was used. The total number of leaf cells advanced

Table 2
Errors and convergence orders in the x-component of the velocity for the four-way vortex merging problem

Domain	L_2						
	$L=6$	O_2	$L=7$	O_2	$L=8$	O_2	$L=9$
Uniform	2.61e−2	1.31	1.05e−2	1.97	2.68e−3	2.11	6.19e−4
Circle	2.61e−2	1.33	1.04e−2	1.96	2.68e−3	1.98	6.81e−4
Adaptive	2.66e−2	1.35	1.04e−2	2.07	2.47e−3	2.05	5.96e−4
Domain	L_∞						
	$L=6$	O_∞	$L=7$	O_∞	$L=8$	O_∞	$L=9$
Uniform	4.46e−1	1.25	1.87e−1	1.95	4.84e−2	1.85	1.34e−2
Circle	4.49e−1	1.27	1.86e−1	1.94	4.85e−2	1.87	1.33e−2
Adaptive	4.45e−1	1.26	1.86e−1	1.94	4.86e−2	1.83	1.37e−2

Table 3
Timings for uniform, circle and adaptive grids for the four-way vortex merging problem

	CPU Time		Cells advanced
	Total (s)	μs/cell	Number
Uniform, $L = 8$	1486	167	8,912,896
Circle, $L = 8$	166	222	746,368
Adaptive, $L = 8$	117	286	409,632
Uniform, $L = 9$	13,034	166	78,643,200
Circle, $L = 9$	1024	183	5,608,960
Adaptive, $L = 9$	764	326	2,342,200

Table 4
Errors and convergence rates for the x-component of the velocity

	All cells			Full level 5 cells		
	5–6	Rate	6–7	5–6	Rate	6–7
L_1	2.66e − 4	1.81	7.60e − 5	2.41e − 4	1.85	6.69e − 5
L_2	5.83e − 4	1.44	2.15e − 4	5.36e − 4	1.49	1.91e − 4
L_∞	5.05e − 3	0.89	2.72e − 3	3.77e − 3	0.93	1.98e − 3

for the whole calculation is given as well as the corresponding average speed. For $L = 8$, a speedup of about nine is obtained when using the statically refined "circle" grid, and 13 when using the adaptive technique. Both the "circle" and adaptive discretisations are notably slower (per cell) than the uniform discretisation. This is mainly due to the interpolations necessary to compute the pressure gradient at coarse/fine cell boundaries when solving the Poisson equation (Fig. 4). It is also interesting to note that the CPU times

Table 5
Errors and convergence rates for the y-component of the velocity

	All cells			Full level 5 cells		
	5–6	Rate	6–7	5–6	Rate	6–7
L_1	2.75e − 4	1.95	7.11e − 5	2.31e − 4	2.16	5.17e − 5
L_2	7.09e − 4	1.32	2.84e − 4	6.76e − 4	1.59	2.25e − 4
L_∞	7.47e − 3	1.05	3.60e − 3	5.98e − 3	1.07	2.85e − 3

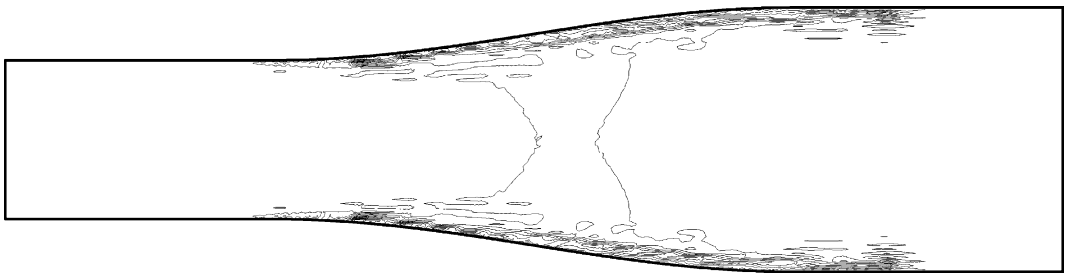

Fig. 22. Contour plot of the error on the x-component of the velocity estimated for a solution with $L = 6$ levels of refinement.

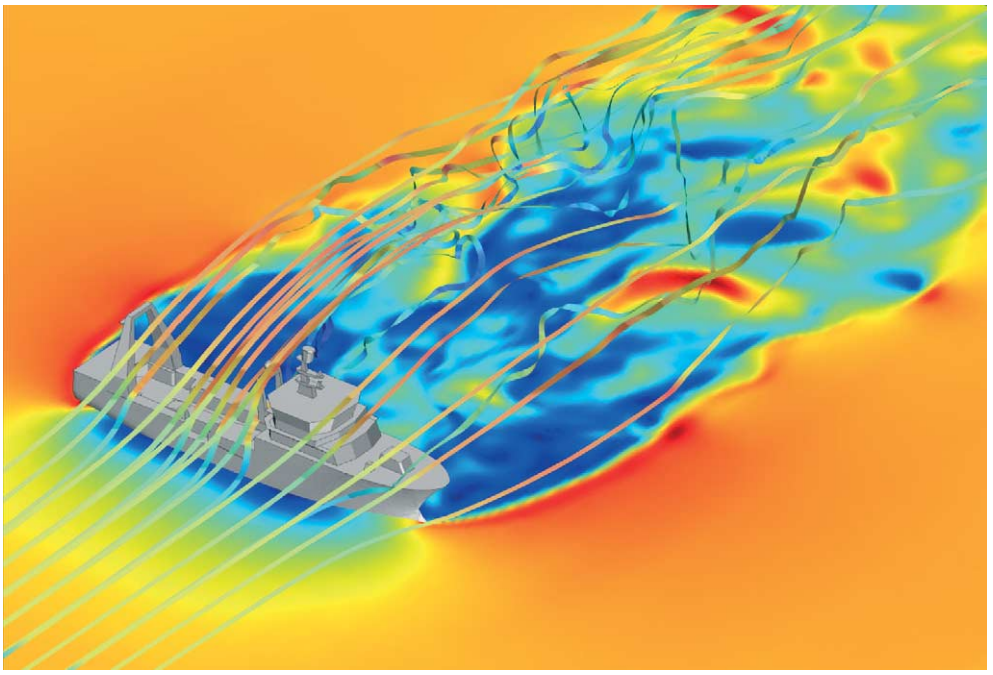

Fig. 23. Airflow around RV Tangaroa. The stream ribbons and cross-section at sea level are coloured according to the norm of the velocity.

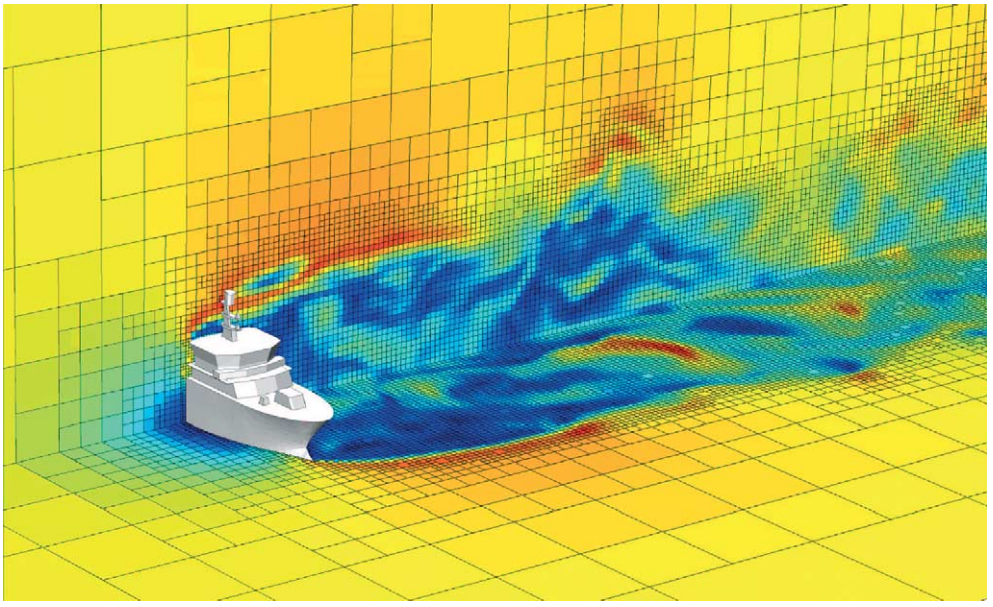

Fig. 24. Adaptive mesh. The horizontal and vertical cross-sections illustrate the three-dimensional adaptive octree.

obtained are very close to those reported by Almgren et al. [43] for the same problem (keeping in mind that they used a Cartesian AMR technique on a DEC Alpha computer and solved a viscous flow).

To demonstrate the convergence properties of the method in the presence of solid boundaries, we use a test case initially presented by Almgren et al. [15]. A diverging channel is constructed in a 4×1 domain by restricting the fluid flow through the curves y_{top} and y_{bot}, defined as

$$y_{\text{bot}} = \begin{cases} y_1 & \text{if } 0 \leqslant x \leqslant 1, \\ y_2 + 0.5(y_1 - y_2)(1 + \cos(\frac{\pi}{2}(x-1)) & \text{if } 1 < x < 3, \\ y_2 & \text{if } 3 \leqslant x \leqslant 4 \end{cases}$$

and

$$y_{\text{top}} = 1 - y_{\text{bot}},$$

with $y_1 = 0.2$ and $y_2 = 10^{-6}$. Neumann boundary conditions for the pressure are set at the inlet ($x = 0$) and at the solid boundaries. A fixed unity inflow velocity is set at the inlet and simple outflow boundary conditions at the outlet (the pressure and the gradients of all the components of the velocity are set to zero at $x = 4$). The simulations are ran to $t = 1$ using a CFL of 0.8. Three simulations are performed on uniform grids with $L = 5, 6$ and 7 levels of refinement. Tables 4 and 5 show the errors and convergence rates obtained. As in [15] we calculate errors both on the full domain (''All cells'') and on the part of the domain covered by cells at level 5 entirely contained within the fluid (''Full level 5 cells''). Columns labelled ''5–6'' give the error computed on the mesh with 5 levels of refinement using the mesh with 6 levels of refinement as reference (and similarly for columns labelled ''6–7''). For both components of the velocity close to first-order convergence is obtained for the L_∞ norm and close to second-order convergence for the L_1 norm, as expected from a solution globally second-order accurate but first-order accurate at the boundaries. Fig. 22 confirms that the error is concentrated near solid boundaries. The maximum error in either component is small (less than one percent of the magnitude of the velocity).

Finally, we present an application of the three-dimensional version of the code to a practical engineering-type problem. The air flow around the vessel RV Tangaroa of the National Institute of Water and Atmospheric Research has been simulated by solving the 3D time-dependent incompressible Euler equations around a CAD model. Fig. 23 is a snapshot in time of the developed turbulent flow. Wind is coming at a right angle from the right of the vessel. The stream ribbons and cross-section at sea level are coloured according to the norm of the velocity. The spatial resolution is about 50 cm near the ship and is adapted dynamically (using the vorticity criterion) down to a minimum scale of one meter elsewhere in the flow. The resulting mesh is composed of about 350,000 leaf cells in established regime. Fig. 24 shows a vertical and horizontal cross-sections through the adapted octree mesh for the same timestep. We are in the process of comparing these results to experimental measurements, which will be the subject of a future publication.

9. Conclusion

The combination of a quad/octree discretisation, an approximate projection method, a multigrid Poisson solver and a volume-of-fluid embedded description of solid boundaries proves to be a feasible and efficient technique for the numerical solution of the time-dependent incompressible Euler equations. This approach differs from the classical Cartesian AMR technique [7,9,12] by treating the connection between levels of refinement at the cell operator level rather than through boundary conditions between refined patches. These operators can be designed to be spatially second-order accurate and to use consistent (conservative) flux estimations at coarse/fine boundaries. This fine-grained description allows almost full flexibility in the placement and shape of refined regions. Moreover, the refinement and coarsening process is naturally

implemented by the quad/octree structure and does not need specialised algorithms for grid generation [44]. The mesh adaptation can thus be performed for every timestep with minimum overhead.

The price to pay for this flexibility is the loss of the array-based, cache- and access-efficient structured grids which are at the core of the Cartesian AMR technique. While more thorough investigation would be necessary, we show that similar performances to AMR can be achieved using our quad/octree approach. Moreover, we believe that in the case of small and complicated structures (such as interfaces between fluids or shocks) the flexibility of this approach can more than compensate for this overhead (given that a Cartesian AMR technique would require a large number of refined patches to cover the small structures, leading to substantial overheads in boundary conditions and most probably to the loss of cache-efficiency).

Future developments include extension to the incompressible variable-density Navier–Stokes equations and interfacial flows, using VOF [45] and marker techniques [46]. Using sub-cycling in time on different levels of refinement [43] would also be a useful extension of the algorithm presented.

Finally, by providing an open source version of the code which can be freely redistributed and modified [25], we hope to encourage research and collaboration in this field.

Acknowledgements

This work was funded by the New Zealand Foundation for Research, Science and Technology and by the Marsden Fund of the Royal Society of New Zealand.

References

[1] A.J. Chorin, Numerical solution of the Navier–Stokes equations, Math. Comp. 22 (1968) 745–762.
[2] R. Peyret, T.D. Taylor, Computational Methods for Fluid Flow, Springer, New York/Berlin, 1983.
[3] A. Brandt, Guide to Multigrid Development, Multigrid Methods, Springer, Berlin, 1982.
[4] J. Wesseling, An Introduction to Multigrid Methods, Wiley, Chichester, 1992.
[5] J.B. Bell, P. Colella, H.M. Glaz, A second-order projection method for the incompressible Navier–Stokes equations, J. Comput. Phys. 85 (1989) 257–283.
[6] J.B. Bell, D.L. Marcus, A second-order projection method for variable density flows, J. Comput. Phys. 101 (1992) 334–348.
[7] M.J. Berger, J. Oliger, Adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 53 (1984) 484–512.
[8] W.J. Coirier, An adaptively-refined, Cartesian, cell-based scheme for the Euler and Navier–Stokes equations, Ph.D. Thesis, NASA Lewis Research Center, Cleveland, OH, USA, October 1994.
[9] L.H. Howell, J.B. Bell, An adaptive mesh projection method for viscous incompressible flow, SIAM J. Sci. Comput. 18 (4) (1997) 996–1013.
[10] A.M. Khokhlov, Fully threaded tree algorithms for adaptive refinement fluid dynamics simulations, J. Comput. Phys. 143 (2) (1998) 519–543.
[11] M.L. Minion, A projection method for locally refined grids, J. Comput. Phys. 127 (1996) 158–178.
[12] A.S. Almgren, J.B. Bell, W.G. Szymczak, A numerical method for the incompressible Navier–Stokes equations based on an approximate projection, SIAM J. Sci. Comput. 17 (1996) 358–369.
[13] M. Berger, R. LeVeque, A rotated difference scheme for Cartesian grids in complex geometries, in: AIAA 10th Computational Fluid Dynamics Conference, Honolulu, Hawaii, 1991, pp. 1–7.
[14] J.J. Quirk, An alternative to unstructured grids for computing gas dynamics flows around arbitrarily complex two-dimensional bodies, Comput. Fluids 23 (1994) 125–142.
[15] A.S. Almgren, J.B. Bell, P. Colella, T. Marthaler, A Cartesian grid projection method for the incompressible Euler equations in complex geometries, SIAM J. Sci. Comput. 18 (1997) 1289–1309.
[16] T. Ye, R. Mittal, H.S. Udaykumar, W. Shyy, An accurate Cartesian grid method for viscous incompressible flows with complex immersed boundaries, J. Comp. Phys. 156 (1999) 209–240.
[17] D. Calhoun, R.J. LeVeque, Solving the advection–diffusion equation in irregular geometries, J. Comp. Phys. 156 (2000) 1–38.
[18] C.S. Peskin, Flow patterns around heart valves: a numerical method, J. Comp. Phys. (1972) 10–252.
[19] E.M. Saiki, S. Biringen, Numerical simulations of a cylinder in a uniform flow: application of a virtual boundary method, J. Comp. Phys. 123 (1996) 450–465.

[20] E.A. Fadlun, R. Verzicco, P. Orlandi, J. Mohd-Yusof, Combined immersed-boundary finite-difference methods for three-dimensional complex flow simulations, J. Comp. Phys. 161 (2000) 35–60.
[21] D.F. Martin, An adaptive cell-centered projection method for the incompressible Euler equations, Ph.D. Thesis, University of California, Berkeley, 1998.
[22] D.F. Martin, P. Colella, A cell-centered adaptive projection method for the incompressible euler equations, J. Comp. Phys. 163 (2000) 271–312.
[23] H. Johansen, P. Colella, A Cartesian grid embedded boundary method for poisson's equation on irregular domains, J. Comp. Phys. 147 (1998) 60–85.
[24] M.S. Day, P. Colella, M.J. Lijewski, C.A. Rendleman, D.L. Marcus, Embedded boundary algorithms for solving the Poisson equation on complex domains, Technical Report LBNL-41811, Lawrence Berkeley National Laboratory, May 1998.
[25] S. Popinet, The Gerris flow solver. Available from http://gfs.sourceforge.net.
[26] H. Samet, Applications of Spatial Data Structures, Addison-Wesley, Reading, MA, 1989.
[27] R. Gonzalez, P. Wintz, Digital Image Processing, Addison-Wesley, Reading, MA, 1987.
[28] M.J. Berger, M.J. Aftosmis, Aspects (and aspect ratios) of Cartesian mesh methods, in: Proceedings of the 16th International Conference on Numerical Methods in Fluid Dynamics, Springer, Arcachon, France, 1998.
[29] L. Balmelli, J. Kovačević, M. Vetterli, Quadtrees for embedded surface visualization: constraints and efficient data structures, in: Proceedings of IEEE International Conference on Image Processing, vol. 2, 1999, pp. 487–491.
[30] H. Edelsbrunner, E.P. Mücke, Simulation of simplicity: a technique to cope with degenerate cases in geometric algorithms, ACM Trans. Graph 9 (1) (1990) 66–104.
[31] K.L. Clarkson, Safe and effective determinant evaluation, in: Proceedings of the 31st IEEE Symposium on Foundations of Computer Science, Pittsburgh, PA, 1992, pp. 387–395.
[32] V.J. Milenkovic, Robust polygon modeling, Comput. Aided Des. 25 (9) (1993) 546–566.
[33] J.R. Shewchuk, Adaptive precision floating-point arithmetic and fast robust geometric predicates, Technical Report, School of Computer Science, Carnegie Mellon University, Pittsburgh, Pennsylvania, May 1996.
[34] S. Popinet, The GNU Triangulated Surface Library. Available from http://gts.sourceforge.net.
[35] M.J. Aftosmis, M.J. Berger, J.E. Melton, Robust and efficient Cartesian mesh generation for component-based geometry, Technical Report AIAA-97-0196, US Air Force Wright Laboratory, 1997.
[36] D.L. Brown, R. Cortez, M.L. Minion, Accurate projection methods for the incompressible Navier–Stokes equations, J. Comput. Phys. 168 (2001) 464–499.
[37] W. Noh, Cel: a time-dependent, two-space-dimensional, coupled Eulerian–Lagrangian code, Fundam. Meth. Hydrodyn. (Meth. Comput. Phys.) 3 (1964) 117–179.
[38] M.F. Lai, A projection method for reacting flows in the zero Mach number limit, Ph.D. Thesis, University of California at Berkeley, 1993.
[39] J.C. Strikwerda, Finite difference methods for the Stokes and Navier–Stokes equations, SIAM J. Sci. Stat. Comput. 5 (1984) 56–67.
[40] W.J. Rider, Approximate projection methods for incompressible flows: implementation, variants and robustness, Technical Report LA-UR-2000, Los Alamos National Laboratory, 1995.
[41] A.S. Almgren, J.B. Bell, W.Y. Crutchfield, Approximate projection methods: Part I. Inviscid analysis, SIAM J. Sci. Comput. 22 (4) (2000) 1139–1159.
[42] M.J. Berger, P. Colella, Local adaptive mesh refinement for shock hydrodynamics, J. Comput. Phys. 82 (1989) 64–84.
[43] A.S. Almgren, J.B. Bell, P. Colella, L.H. Howell, M.L. Welcome, A conservative adaptive projection method for the variable density incompressible Navier–Stokes equations, J. Comp. Phys. 142 (1998) 1–46.
[44] M.J. Berger, I. Rigoutsos, An algorithm for point clustering and grid generation, IEEE Trans. Syst., Man Cybernet. 21 (1991) 1278–1286.
[45] D. Gueyffier, A. Nadim, J. Li, R. Scardovelli, S. Zaleski, Volume of fluid interface tracking with smoothed surface stress methods for three-dimensional flows, J. Comp. Phys. 152 (1998) 423–456.
[46] S. Popinet, S. Zaleski, A front tracking algorithm for the accurate representation of surface tension, Int. J. Numer. Meth. Fluids 30 (1999) 775–793.